Mikrokomputery

Henryk Małysiak
Bolesław Pochopień
Eugeniusz Wróbel

Mikrokomputery klasy IBM PC

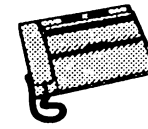
OPTIMUS®

KOMPUTERY NAJLEPSZEJ JAKOŚCI
W KAŻDEJ SYTUACJI

286

386 SX

?

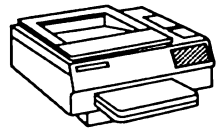

FAXY

18 MIESIĘCY GWARANCJI

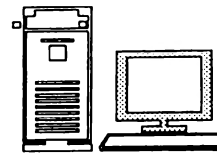
45 PUNKTÓW SERWISOWYCH W POLSCE, CZECHOSŁOWACJI I HOLANDII

KOMPATYBILNY Z IBM W 100%

SIECI NOWELL


KOPIARKI

UNIX


486

Mepior® JEDYNY AUTORYZOWANY DYSTRYBUTOR W EUROPIE WSCHODNIEJ

SANYO KOMPUTERY, FAXY, KOPIARKI

Microsoft® AUTORYZOWANY DYSTRYBUTOR

OPTYMALNA JAKOŚĆ Z NASZYMI CENAMI

ADRESY PUNKTÓW SPRZEDAŻY ORAZ CENY ZNAJDZIESZ W DWUTYGODNIKU PC kurier

OPTIMUS™

OPTIMUS®

Nowy Sącz, ul. Barbackiego 10
tel. (018)227-97, fax. (018)221-85
tlx. 326450 optim pl

Mikrokomputery klasy IBM PC

Mikrokomputery

Komitet Redakcyjny

Sekretarz Wojciech Cellary
Zuzanna Grzejszczak
Andrzej Kobus
Przewodniczący Romuald Marczyński
Piotr Misiurewicz
Wojciech Nowakowski
Maciej Stolarski
Halina Tempczyk
Józef Winkowski
Jan Zabrodzki

Henryk Małysiak
Bolesław Pochopień
Eugeniusz Wróbel

Mikrokomputery klasy IBM PC

Wydawnictwa Naukowo-Techniczne
Warszawa

Opiniodawca *Ryszard Goczyński*
Redaktor *Małgorzata Marczuk*
Przygotowała do druku *Ewa Eckhardt*
Projekt serii *Juliusz Rybicki*
Okładkę i strony tytułowe projektował *Andrzej Pilich*

681.3

Książka zawiera przegląd zagadnień związanych z mikrokomputerami typu IBM PC, przy czym uwagę skupiono przede wszystkim na sprzęcie, a także na bezpośrednim programowaniu mikroprocesorów użytych w tych komputerach. Dość szczegółowo omówiono podstawowe moduły omawianych mikrokomputerów – sterowniki pamięci dyskowych, klawiaturę, moduły grafiki itp. – kładąc nacisk zwłaszcza na jednostki centralne. Opis wzbogacono licznymi schematami ideowymi i komentarzami, a także danymi o charakterze katalogowym.

Książka jest przeznaczona dla inżynierów, studentów, techników oraz dla zainteresowanych użytkowników sprzętu typu IBM PC/XT lub AT.

© Copyright by Wydawnictwa Naukowo-Techniczne
Warszawa 1992

All rights reserved
Printed in Poland

ISBN 83-204-1461-X

Spis treści

1. Wprowadzenie / 7

2. Mikroprocesory 16–bitowe / 8

2.1. Mikroprocesory 8086 i 8088 / 8
2.2. Mikroprocesory 80186 i 80188 / 34
2.3. Mikroprocesor 80286 / 44
2.4. Koprocesory arytmetyczne 8087 i 80287 / 74
2.4.1. Słowa liczbowe / 74
2.4.2. Architektura / 78
2.4.3. Lista rozkazów / 81

3. Mikrokomputery 16–bitowe / 86

3.1. Mikrokomputer z mikroprocesorem 8088 / 86
3.1.1. Jednostka centralna mikrokomputera IBM PC/XT / 86
3.1.2. Podsystem pamięci RAM / 95
3.1.3. Podsystem pamięci ROM / 99
3.1.4. Podsystem we–wy / 103
3.2. Mikrokomputer z mikroprocesorem 80286 / 114
3.2.1. Jednostka centralna mikrokomputera IBM PC/AT / 114
3.2.2. Podsystem pamięci ROM / 117
3.2.3. Podsystem pamięci RAM / 117
3.2.4. Podsystem we–wy / 120
3.3. Moduł klawiatury / 128
3.4. Moduł we–wy / 131
3.5. Sterowniki pamięci dyskowych / 134
3.6. Moduły grafiki / 141
3.7. Inne Moduły / 159

4. Wybrane zagadnienia oprogramowania / 162

4.1. Rozkazy mikroprocesora 80286 /162
4.1.1. Zbiór rozkazów mikroprocesorów 8086/8088 / 162
4.1.2. Rozszerzenia procesora 80286 / 170
4.2. Praca w trybie adresowania wirtualnego
Przełączanie zadań / 184
4.3. Wielozadaniowy system operacyjny OS/2 / 204

5. Przykładowe konfiguracje mikrokomputerów / 213

Literatura / 218

1. Wprowadzenie

Mikrokomputery 16–bitowe IBM PC/XT (ang. *extended technology)* i IBM PC/AT (ang. *advanced technology*) dość powszechnie opanowały rynek światowy oraz krajowy. Występuje więc duże zapotrzebowanie na wszelkiego rodzaju informacje związane z ich budową i oprogramowaniem.

Autorzy książki w pewnym stopniu wychodzą naprzeciw tym problemom, głównie w zakresie konstrukcji tych mikrokomputerów i dodatkowych modułów (kart) rozszerzających ich możliwości funkcjonalne oraz możliwości współpracy z różnorodnymi urządzeniami zewnętrznymi.

W rozdziale drugim szczegółowo omówiono mikroprocesory 16–bitowe 8086, 8088, 80186, 80188, 80286 firmy Intel oraz koprocesory 8087 i 80287.

Szczególnie dużo uwagi poświęcono mikroprocesorowi 16–bitowemu 8088 oraz mikroprocesorowi 80286, gdyż są one stosowane jako procesory główne w mikrokomputerach odpowiednio IBM PC/XT i IBM PC/AT.

Rozdział trzeci stanowi istotną część książki i dotyczy bezpośrednio konstrukcji mikrokomputerów IBM PC. Przedstawiono w nim funkcje, działanie oraz budowę układów i modułów, tworzących tzw. *płyty główne* tych mikrokomputerów oraz podstawowych modułów rozszerzających, takich jak: sterowniki dysków, moduły grafiki lub moduły akwizycji obrazu.

Rozdział czwarty poświęcono wybranym zagadnieniom oprogramowania mikrokomputerów z mikroprocesorami firmy Intel, że szczególnym uwzględnieniem mikroprocesora 80286 pracującego w trybie adresowania wirtualnego (rozkazy związane z tym trybem pracy, problem przełączania zadań i inne). W rozdziale zawarto również wprowadzenie do wielozadaniowego systemu operacyjnego OS/2.

W rozdziale piątym przedstawiono typowe konfiguracje mikrokomputerów IBM PC/XT i AT oraz konfiguracje przeznaczone do realizacji określonych zadań na przykładzie wspomagania prac wydawniczych i wspomagania projektowania układów cyfrowych.

Tytuł książki "Mikrokomputery klasy IBM PC" wskazuje, że dotyczy ona nie tylko mikrokomputerów IBM PC/XT/AT oraz zgodnych z nimi, ale i innych 16–bitowych, o zbliżonych możliwościach funkcjonalnych i rozwiązaniach układowych.

2. Mikroprocesory 16–bitowe

2.1. Mikroprocesory 8086 i 8088

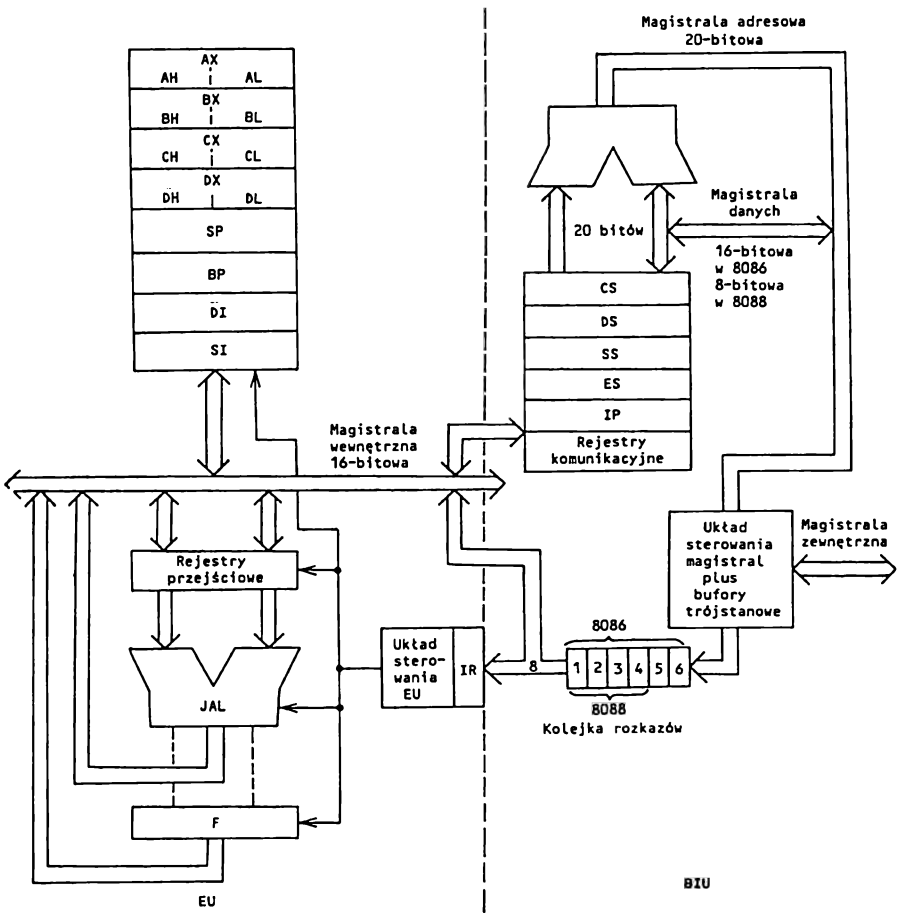
Mikroprocesory 8086 i 8088 należą do jednych z pierwszych, szeroko rozpowszechnionych mikroprocesorów 16–bitowych. Mają one wiele cech wspólnych z mikroprocesorami 8080, 8085; są jednak znacznie bardziej efektywne ze względu na zastosowanie 16–bitowej architektury i znacznie bogatszej listy rozkazów, którą można rozszerzyć o rozkazy specjalizowanego koprocesora (np. arytmetycznego 8087) oraz możliwość pracy w systemach jedno i wielomikroprocesorowych. Mikroprocesory 8086 i 8088 mają prawie jednakowe architektury i identyczne listy rozkazów. Istotna różnica polega na tym, że mikroprocesor 8086 ma 16–bitową, a mikroprocesor 8088 8–bitową *magistralę danych*. Dalsze rozważania dotyczą obu tych procesorów, a niewielkie różnice występujące w ich budowie są wyraźnie podkreślone. Pozostałe ważniejsze parametry tych mikroprocesorów to:

- 1–megabajtowa (MB) przestrzeń adresowa pamięci (20 linii adresowych);
- 64–kilobajtowa (KB) przestrzeń adresowa we–wy;
- podział na dwa niezależne układy umożliwiające równoległe wykonywanie bieżącego rozkazu i pobieranie rozkazów następnych;
- dwa tryby pracy (minimalny, maksymalny);
- możliwość współpracy z 8–bitowymi programowanymi układami we–wy firmy Intel;
- częstotliwość impulsów zegarowych 4, 5, 8, 10 MHz – zależnie od wersji wykonania;
- jedno napięcie zasilania $+5V \pm 0{,}5$.

Strukturę wewnętrzną mikroprocesora 8086 (8088) przedstawiono na rys. 2.1. Wyróżnić w niej można

- układ sterowania magistrali BIU (ang. *bus interface unit*);
- układ wykonawczy EU (ang. *execution unit*).

Do zadań *układu sterowania magistrali* należy wyznaczanie adresu pamięci, pobieranie kodów rozkazów do wewnętrznej pamięci RAM typu FIFO (ang. *first in first out*), tworzącej 6–bajtową (w mikroprocesorze 8088 – 4–bajtową) kolejkę oraz pobieranie i przesyłanie z (do) pamięci argumentów na żądanie układu EU.



Rys. 2.1. Struktura wewnętrzna mikroprocesora 8086 (8088)

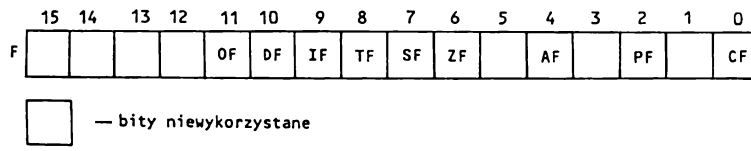
Podstawowym zadaniem *układu wykonawczego* EU jest dekodowanie i wykonywanie rozkazów pobieranych z kolejki. Układy te w większości przypadków mogą pracować niezależnie, co umożliwia równoległe w czasie wykonywanie rozkazu bieżącego w układzie EU i pobieranie przez BIU następnych rozkazów do kolejki. Występują tu więc pewne cechy architektury potokowej (ang. *pipelined*). Zapewnia to bardziej efektywne czasowo wykorzystanie zewnętrznej magistrali w mikroprocesorze 8086 (8088), niż w mikroprocesorach 8080 i Z80, w których kolejne fazy bieżącego i następnych cykli rozkazowych są wykonywane szeregowo.

BIU organizuje pobranie kolejnego kodu rozkazu z pamięci programu po przesłaniu do EU każdych dwóch bajtów z kolejki. Jeżeli wykonywany rozkaz wymaga np. pobrania z pamięci argumentu (argumentów), to EU sygnalizuje ten fakt układowi BIU, który wstrzymuje pobieranie kolejnego rozkazu i inicjuje cykl odczytu argumentu.

Gdy EU realizuje rozkaz (skoku lub wywołania), powodujący rozgałęzienie programu, wówczas następuje zerowanie kolejki i BIU zaczyna pobierać do niej rozkazy od adresu, do którego było przeniesione sterowanie.

Układ wykonawczy zawiera 16-bitową *jednostkę arytmetyczno-logiczną* JAL (ang. *ALU – arithmetic-logic unit*), układ sterowania z rejestrem rozkazów i zestaw rejestrów dostępnych w programie. Wszystkie rejestry są 16-bitowe. Należą do nich rejestry robocze AX, BX, CX i DX, wskaźnik stosu SP, wskaźnik bazy BP, rejestry indeksowe źródła SI (ang. *source index*) i przeznaczenia DI (ang. *destination index*) ciągów łańcuchów danych oraz rejestr znaczników F (ang. *flags*). Służą one do przechowywania danych i adresów. Do rejestrów roboczych można się odwoływać jako do rejestrów 16-bitowych AX, BX, CX, DX lub jako do rejestrów 8-bitowych AH, BH, CH, DH i AL, BL, CL, DL. Część tych rejestrów ma swoje odpowiedniki w mikroprocesorach ośmiobitowych 8080(85) i Z80.

Rejestr AL – odpowiednik akumulatora A; BX – *rejestr bazowy*, służący do adresowania – odpowiednik pary rejestrów HL; CX – *rejestr zliczający*, służący między innymi do zliczania liczby powtórzeń pętli programowych – odpowiednik pary rejestrów BC; DX – *rejestr danych*, służący m.in. do adresowania układów we-wy – odpowiednik pary rejestrów DE. Pozostałe rejestry są 16-bitowe. *Rejestr znaczników* F zawiera 9 znaczników. Ich rozmieszczenie przedstawiono na rys. 2.2. Można je podzielić na *znaczniki stanu*, zależne od wyniku ostatniej operacji w JAL, i *znaczniki sterujące.*



Rys. 2.2. Zawartość rejestru znaczników F

Znaczniki stanu to:

- **A** (ang. *auxiliary carry*) – przeniesienia połówkowego;
- **C** (ang. *carry*) – przeniesienia;
- **O** (ang. *overflow*) – przepełnienia (dla operacji arytmetycznych);
- **S** (ang. *sign*) – znaku;
- **P** (ang. *parity*) – parzystości;
- **Z** (ang. *zero*) – zera;

Znaczniki sterujące to:

- **D** (ang. *direction*) – znacznik kierunku zmian zawartości rejestrów indeksowych źródła i przeznaczenia przy operacjach na łańcuchach danych; Przy D = 0 zwiększa się zawartość rejestrów SI i DI o 1 po przesłaniu każdego bajtu (przy przesłaniach słów 8-bitowych) i o 2 po przesłaniu każdego słowa (przy przesłaniach słów 16-bitowych). Przy D = 1 zmniejsza się zawartość tych rejestrów (patrz p. 4.1.1.).
- **I** (ang. *interrupt enable*) – znacznik wskazujący, czy wejście przerwania maskowanego jest odblokowane (I = 1), czy zablokowane (I = 0);

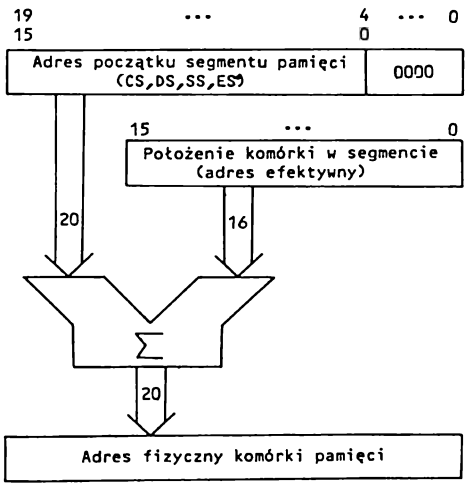
T (ang. *trap*) – znacznik pracy krokowej (pułapki). T = 1 ustawia procesor w tryb pracy krokowej.

W skład układu BIU wchodzą między innymi 4 rejestry segmentów pamięci (ang. *segment register*) zawierające adresy początków segmentów pamięci, sumator do wyznaczania adresu fizycznego pamięci, wskaźnik rozkazów IP (ang. *instruction pointer*) zwany również wskaźnikiem programu i wspomniana już kolejka. BIU jest odpowiedzialny za przesłania i ich organizację między mikroprocesorem a pamięcią lub układami we–wy, stosownie do żądań układu EU i kolejki.

Adresowanie pamięci

Zorganizowaną bajtowo pamięć mikroprocesorów 8086/88 dzieli się na segmenty. Każdy bajt pamięci ma dwa rodzaje adresów: logiczny i fizyczny. *Adres logiczny* składa się z *adresu początku segmentu*, zawartego w jednym z czterech rejestrów segmentów CS, DS, SS i ES i *adresu efektywnego* (ang. *offset*), wskazującego położenie komórki w segmencie, zawartego w jednym z pozostałych przeznaczonych do tego celu rejestrów lub wyliczonego w układzie EU zgodnie z trybem adresowania wykonywanego rozkazu. Adresy te są 16–bitowe.

Na podstawie adresu logicznego w układzie BIU wyznacza się 20–bitowy *adres fizyczny* bajtu pamięci, dodając adres efektywny do adresu początku segmentu przesuniętego o 4 pozycje w lewo (pomnożonego przez 16) – rys. 2.3. Przestrzeń bezpośredniego adresowania pamięci zawiera więc 1048576 bajtów (1MB).



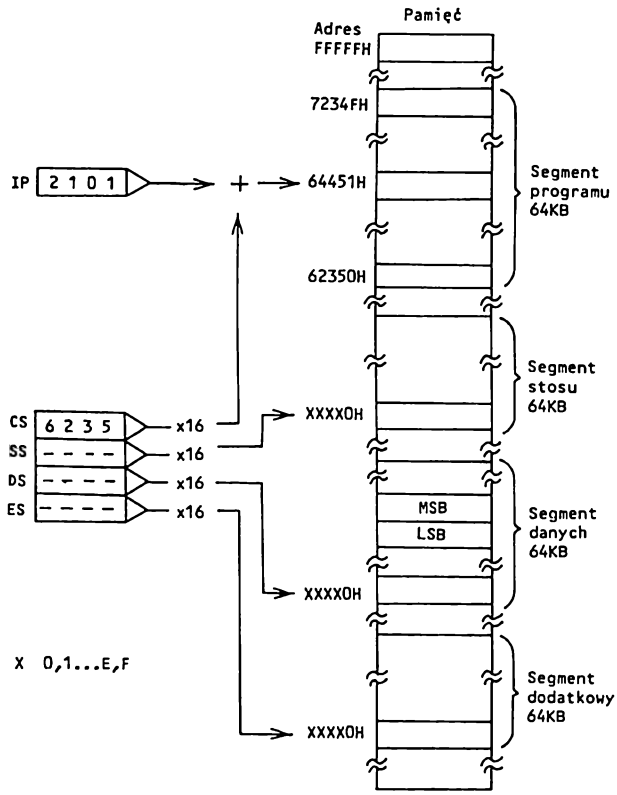
Rys. 2.3. Metoda wyznaczania adresu fizycznego

Podczas wykonywania programu, za pośrednictwem BIU można określić dostęp do jednego z czterech segmentów pamięci

– segmentu programu (kodu) CS (ang. *code segment*);

- segmentu danych **DS** (ang. *data segment*);
- segmentu stosu **SS** (ang. *stack segment*);
- segmentu dodatkowego **ES** (ang. *extra segment*).

Adresy początków tych segmentów deklaruje się w programie przez zapisanie odpowiednich wartości do określających je 16-bitowych rejestrów CS, DS, SS i ES. Maksymalna pojemność segmentu wynosi 64 KB. Początki segmentów mogą znajdować się w dowolnym miejscu przestrzeni adresowej pamięci (rys. 2.4). Segmenty mogą zachodzić na siebie, a w szczególnym przypadku (gdy ich adresy początkowe są jednakowe) mogą się pokrywać. Możliwość programowej zmiany zawartości rejestrów segmentów zapewnia łatwe i dynamiczne przemieszczanie programów w pamięci.



Rys. 2.4. Przykładowe rozmieszczenie segmentów w pamięci

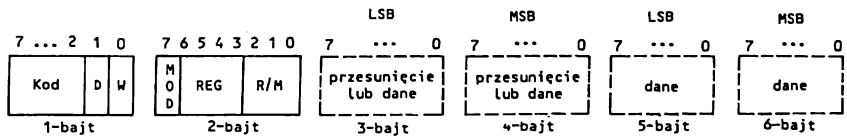
Typowe przeznaczenie rejestrów segmentów:

- *rejestr* **CS** wskazuje początek segmentu pamięci zawierającego program. Położenie rozkazu w tym segmencie określa zawartość wskaźnika rozkazów IP;

- *rejestr* **DS** wskazuje początek segmentu pamięci zawierającego dane (argumenty). Położenie argumentu w tym segmencie określa adres efektywny argumentu, zależny od sposobu adresowania (sposoby adresowania omówimy dalej). Używa się go również do wyznaczania adresu fizycznego w operacjach na ciągach słów, których położenie w segmencie określa rejestr SI;
- *rejestr* **SS** wskazuje początek segmentu pamięci, stanowiącej stos programowy. Położenie danej w tym segmencie, przy standardowych operacjach zapisu i odczytu stosu (*PUSH*, *POP*), określa zawartość wskaźnika stosu SP, a przy pozostałych operacjach możliwych do wykonania na stosie – zawartość wskaźnika bazowego BP. Operacje wykonywane na stosie dotyczą słów 16–bitowych. Stos "rośnie" w kierunku malejących adresów wewnątrz segmentu stosu, zatem rozkaz *PUSH* umieszczający daną na wierzchołku stosu powoduje zmniejszenie wskaźnika SP o 2, a rozkaz *POP* pobierający daną ze stosu – zwiększenie wskaźnika SP o 2.
- *rejestr* **ES** wskazuje początek dodatkowego segmentu danych w pamięci. Segment ten często stosuje się do przechowywania danych wspólnych dla kilku programów oraz do wyznaczania adresu fizycznego w operacjach na ciągach słów, których położenie w segmencie określa rejestr DI.

Sposoby adresowania

Rozkazy mikroprocesorów 8086/88 mogą mieć długość od 1 do 6 bajtów. Rozkazami jednobajtowymi są zwykle rozkazy sterujące np. znacznikami lub wykonujące operacje na zawartości określonego rejestru albo na ciągach danych. Rozkazy wielobajtowe mają najczęściej format przedstawiony na rys. 2.5, z tym, że bajty od 3 do 6 mogą występować opcjonalnie, zależnie od zawartości dwóch pierwszych bajtów. Dotyczą one zwykle operacji dwuargumentowych. Sześć najbardziej znaczących bitów pierwszego bajtu rozkazu zawiera jego kod (w niektórych rozkazach część kodu jest zawarta w drugim bajcie rozkazu, w polu REG). Bit b1 pierwszego bajtu, zależnie od rozkazu, oznacza się jako D, S lub V.



Rys. 2.5. Format rozkazu mikroprocesora 8086 (8088)

Symbol **D** (ang. *direction*) określa kierunek przesyłania danych. D = 1 oznacza, że dane są przesyłane do rejestru, wskazanego przez pole REG drugiego bajtu rozkazu, a D = 0 oznacza, że dane są przesyłane do rejestru lub komórki pamięci, wskazanych przez pole R/M.

Symbol **S** należy interpretować łącznie z symbolem W, występującym na najmniej znaczącym bicie pierwszego bajtu rozkazu. Symbol ten występuje w rozkazach z adresowaniem natychmiastowym. SW = 01 oznacza, że trzeci i czwarty bajt rozkazu stanowią 16–bitowy argument; SW = 11 oznacza, że trzeci bajt rozkazu po rozszerzeniu do 16–tu bitów z zachowaniem znaku stanowi argument.

Symbol **V** występuje w rozkazach przesuwania zawartości rejestru lub komórki pamięci i określa liczbę przesunięć. V = 0 oznacza, że liczba przesunięć = 1, a V = 1, że liczbę przesunięć określa zawartość rejestru CL.

Symbol **W** określa, czy operacja będzie wykonywana na słowie 8-bitowym (W = 0), czy na słowie 16-bitowym (W = 1).

Symbol **Z** występuje w jednobajtowym rozkazie REP na najmniej znaczącym bicie i służy do określania liczby powtórzeń wykonywania rozkazu operującego na danych, poprzedzonego tym rozkazem. Przy Z = 0 powtarzanie odbywa się aż do chwili, w której znacznik ZF osiągnie stan 1. Przy Z = 1 powtarzanie odbywa się aż do chwili, w której znacznik ZF osiągnie stan 0.

Pole **REG** wskazuje rejestr, zawierający pierwszy argument w operacjach dwuargumentowych – tabl. 2.1 (w rozkazach z adresowaniem natychmiastowym pole REG zawiera rozszerzenie kodu operacji).

Tablica 2.1

REG lub R/M dla MOD = 11	W = 0	W = 1
0 0 0	AL	AX
0 0 1	CL	CX
0 1 0	DL	DX
0 1 1	BL	BX
1 0 0	AH	SP
1 0 1	CH	BP
1 1 0	DH	SI
1 1 1	BH	DI

Pole **MOD** w drugim bajcie rozkazu określa, czy drugi argument znajduje się w komórce pamięci, czy też w rejestrze – tabl. 2.2 – gdzie: *d* (ang. *displacement*) oznacza przemieszczenie 1-bajtowe, a *dd* – 2-bajtowe.

Tablica 2.2

MOD	
0 0	Drugi argument w pamięci Brak przesunięcia, z wyjątkiem R/M = 110
0 1	Drugi argument w pamięci Adres efektywny obliczany z uwzględnieniem *d*
1 0	Drugi argument w pamięci Adres efektywny obliczany z uwzględnieniem *dd*
1 1	Drugi argument w rejestrze

W pierwszym przypadku sposób wyznaczania adresu efektywnego argumentu zależy od pola R/M – tabl. 2.3. W drugim przypadku pole R/M wskazuje rejestr zawierający argument (patrz tabl. 2.1).

Tablica 2.3

R/M	Adres efektywny		
	MOD = 00	MOD = 01	MOD = 10
0 0 0	BX + SI	BX + SI + d	BX + SI + dd
0 0 1	BX + DI	BX + DI + d	BX + DI + dd
0 1 0	BP + SI	BP + SI + d	BP + SI + dd
0 1 1	BP + DI	BP + DI + d	BP + DI + dd
1 0 0	SI	SI + d	SI + dd
1 0 1	DI	DI + d	DI + dd
1 1 0	dd	BP + d	BP + dd
1 1 1	BX	BX + d	BX + dd

W mikroprocesorach 8086/88 można wyróżnić następujące tryby adresowania argumentu w pamięci (patrz tabl. 2.1. do 2.3.):

- *rejestrowe bezpośrednie* – argument znajduje się w rejestrze 8 lub 16–bitowym wskazanym przez rozkaz;
- *natychmiastowe* – argument jest zawarty w rozkazie;
- *rejestrowe pośrednie* – adres efektywny argumentu jest określony zawartością rejestrów SI, DI, BX;
- *bezpośrednie* – adres efektywny argumentu jest zawarty w rozkazie jako 8 lub 16–bitowe przemieszczenie (*d* lub *dd*);
- *bazowe* – adres efektywny argumentu jest sumą 8 lub 16–bitowego przemieszczenia, zawartego w rozkazie i zawartości jednego z rejestrów bazowych BP lub BX;
- *indeksowe* – adres efektywny jest sumą 8–bitowego (gdy MOD = 01) lub 16–bitowego (gdy MOD = 10) przemieszczenia i zawartości jednego z rejestrów indeksowych DI lub SI;
- *bazowo–indeksowe* – adres efektywny jest sumą zawartości jednego z rejestrów indeksowych DI lub SI i zawartości jednego z rejestrów bazowych BP lub BX;
- *bazowo–indeksowe z przemieszczeniem* – adres efektywny jest sumą zawartości jednego z rejestrów indeksowych DI lub SI, zawartości jednego z rejestrów BX lub BP i przemieszczenia 8 lub 16–bitowego.

Adresowanie układów we–wy

Układy we–wy w mikroprocesorach 8086/88 mogą być adresowane w jednolitej lub rozdzielonej przestrzeni adresowej. W pierwszym przypadku układy we–wy są traktowane przez mikroprocesor jak komórki (bajty) pamięci. W drugim przypadku mikroprocesor zapewnia dwa sposoby adresowania tych układów:

- *bezpośrednie*, odnoszące się do rozkazów we–wy, w których po kodzie operacji występuje 8–bitowy adres układu we–wy (podobnie jak w mikoprocesorach 8080/85 i Z80);
- *pośrednie rejestrowe*, odnoszące się do rozkazów we–wy, których kod operacji wskazuje, że rejestr DX zawiera adres układu we–wy. Jest więc możliwe zaadre-

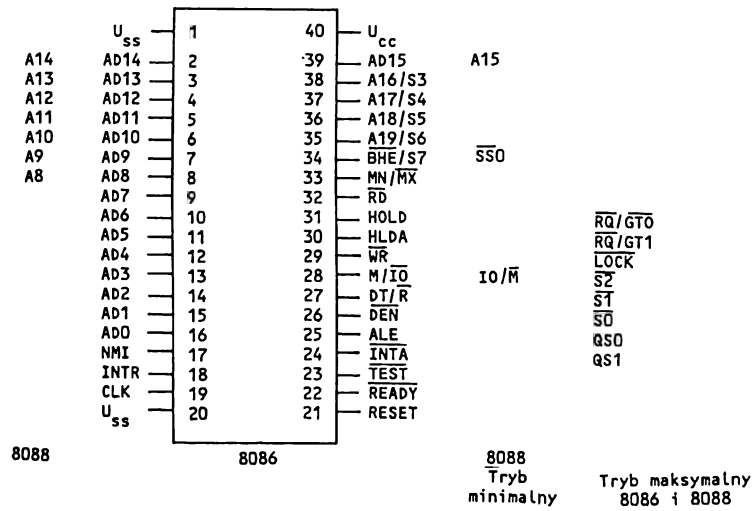
sowanie 65536 układów we-wy operujących na bajtach lub 32768 układów we-wy operujących na słowach 16-bitowych.

Rozdzielona przestrzeń adresowa układów we-wy nie jest segmentowana. Określony w EU adres układu we-wy pojawia się więc bezpośrednio (bez translacji w BIU) na 8 lub 16 mniej znaczących liniach adresowych mikroprocesora, przy czym na bardziej znaczących liniach adresowych pojawiają się zera.

Komunikacja mikroprocesora 8086/88 z otoczeniem

Mikroprocesory 8086/88 mogą pracować w dwóch trybach
- minimalnym (stosowanym zwykle w systemach jednoprocesorowych);
- maksymalnym (stosowanym zwykle w systemach wieloprocesorowych).

Tryb pracy ustala się, podając odpowiedni stan logiczny na wyprowadzenie MN/$\overline{MX}$. Gdy MN/$\overline{MX}$ = 1 – będzie wybrany *tryb minimalny*, a gdy MN/$\overline{MX}$ =0 – *tryb maksymalny*.



Rys. 2.6. Rozmieszczenie wyprowadzeń mikroprocesora 8086 (8088)

Na rysunku 2.6 przedstawiono rozmieszczenie wyprowadzeń mikroprocesorów 8086/88. Wyprowadzeniom od 24 do 31 przydziela się różne funkcje, zależnie od wybranego trybu pracy mikroprocesora. Pozostałe wyprowadzenia mają jednakowe znaczenie w obu trybach pracy.

Wyprowadzenia o jednakowych funkcjach

AD15...AD0 (w mikroprocesorze 8088 – **AD7...AD0**) – dwukierunkowa, trójstanowa, multipleksowana magistrala adresowa i danych. Adres pojawia się na liniach tej magistrali w takcie T_1, a dane w taktach T_2, T_3, T_W (ang. *wait*) i T_4 cyklu magistrali. (W mikroprocesorze 8088 linie adresowe A15...A8 nie są multipleksowane z danymi);

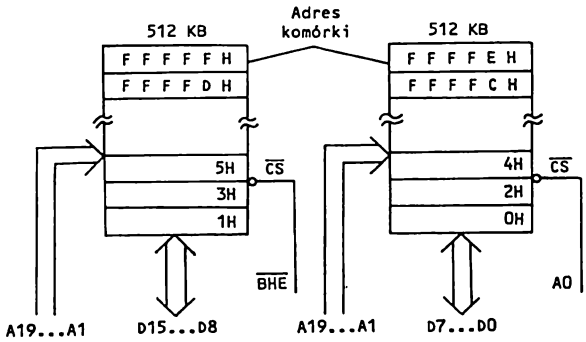
A19/S6, A18/S5, A17/S4, A16/S3 – wyjściowe sygnały trójstanowe. W takcie T_1 sygnały te określają 4 najbardziej znaczące bity adresowe w cyklach pamięci (w cyklach we–wy sygnały te przechodzą do stanu 0). W taktach T_2, T_3, T_w, T_4 sygnały te określają najbardziej znaczące bity statusu S6...S3. Znaczenie ich jest następujące:
S6 – przyjmuje stały stan 0;
S5 – powtarza stan znacznika przerwań IF;
S4, S3 – wskazują, który z rejestrów segmentowych jest używany do adresowania (tabl. 2.4)

Tablica 2.4

S4S3	Rejestr segmentu
00	ES
01	SS
10	CS lub żaden dla cykli we-wy i potwierdzenia przerwania
11	DS

Tablica 2.5

$\overline{BHE}$	AO	Wybrane:
0	0	Słowo 16-bitowe - adres parzysty
0	1	Bardziej znaczący bajt słowa - adres nieparzysty
1	0	Mniej znaczący bajt słowa - adres parzysty
1	1	Pamięć nieaktywna



Rys. 2.7. Organizacja pamięci 16–bitowej

$\overline{BHE}$/S7 – wyjściowy sygnał trójstanowy. W mikroprocesorze 8086 sygnał $\overline{BHE}$ = 0 (ang. *bus high enable*) może pojawić się w takcie T_1 cykli zapisu, odczytu i potwierdzenia przerwania, w celu umożliwienia przesłania bajtu danych bardziej znaczącą połową multipleksowanej magistrali AD15...AD8. Gdy sygnały A0 = 0 i $\overline{BHE}$ = 0, jest możliwe przesłanie między mikroprocesorem a pamięcią (układem we–wy) 16–bitowego słowa w jednym cyklu magistrali (patrz tabl. 2.5 i rys. 2.7). Z rysunku tego wynika, że sygnał $\overline{BHE}$ pełni funkcję sygnału uaktywniającego połowę przestrzeni adresowej, zawierającej komórki pamięci (w przypadku rozdzielonej przestrzeni adresowej) lub komórki pamięci i układy we–wy (w przypadku jednolitej przestrzeni adresowej) o adresach nieparzystych. W taktach T_2, T_3, T_w i T_4 sygnał ten określa bit S7 statusu mikroprocesora (aktualnie niewykorzystywany). W mikroprocesorze 8088 na wyprowadzeniu tym pojawia się sygnał wyjściowy $\overline{SS0}$, który w minimalnym trybie pracy mikroprocesora jest logicznym odpowiednikiem bitu statusu $\overline{S0}$ i wspólnie z sygnałami IO/$\overline{M}$ oraz DT/$\overline{R}$ określa jednoznacznie realizowany cykl magistrali (tabl. 2.6), a w trybie maksymalnym jest zawsze utrzymywany w stanie 1.

Tablica 2.6

IO/$\overline{M}$	DT/$\overline{R}$	$\overline{SSO}$	Rodzaj cyklu magistrali
0	0	0	Cykl pobrania kodu rozkazu
0	0	1	Cykl odczytu pamięci
0	1	0	Cykl zapisu pamięci
0	1	1	Stan „pasywny"
1	0	0	Cykl potwierdzenia przerwania
1	0	1	Cykl odczytu układu wejściowego
1	1	0	Cykl zapisu układu wyjściowego
1	1	1	Cykl potwierdzenia stanu zatrzymania

$\overline{RD}$ – wyjściowy sygnał trójstanowy. Sygnał $\overline{RD}$ = 0 wskazuje, że mikroprocesor realizuje operację odczytu pamięci lub układu we–wy. $\overline{RD}$ = 0 w taktach T_2, T_3 i T_w;

READY – sygnał wejściowy. Sygnał READY = 1 oznacza, że pamięć lub układ we–wy są gotowe (mikroprocesor nie wydłuża cyklu magistrali o dodatkowe takty oczekiwania). Gdy sygnał READY = 0, wówczas po takcie T_3 mikroprocesor przechodzi do stanu oczekiwania WAIT;

INTR – wejściowy sygnał żądania przerwania maskowalnego, aktywny poziomem wysokim;

NMI – wejściowy sygnał żądania przerwania niemaskowalnego, aktywny narastającym zboczem;

TEST – sygnał wejściowy. Stan tego sygnału można testować w programie rozkazem *WAIT*. Gdy $\overline{TEST}$ = 1, wówczas procesor przechodzi do stanu bezczynności i czeka na zmianę stanu sygnału $\overline{TEST}$ z 1 na 0. Gdy sygnał $\overline{TEST}$ = 0 (jeżeli stan 0 jest utrzymywany przez 6 taktów zegarowych), wykonywanie programu jest kontynuowane. Sygnał $\overline{TEST}$ jest synchronizowany narastającym zboczem sygnału CLK;

RESET – wejściowy sygnał zerowania mikroprocesora. W celu poprawnego zerowania sygnał ten powinien być utrzymywany w stanie aktywnym (1) przez co najmniej 4 takty zegarowe. Jest on wewnętrznie synchronizowany;

Ucc – wyprowadzenie +5 V napięcia zasilania;

Uss – wyprowadzenie 0 V napięcia zasilania.

Wyprowadzenia dla *trybu minimalnego*

M/$\overline{IO}$ (**IO/$\overline{M}$** – w mikroprocesorze 8088) – wyjściowy sygnał trójstanowy, umożliwiający rozróżnienie cyklu dostępu do pamięci (M/$\overline{IO}$ = 1 w mikroprocesorze 8086, IO/$\overline{M}$ = 0 w mikroprocesorze 8088) od cykli dostępu do układów we–wy (M/$\overline{IO}$ = 0 w 8086 IO/$\overline{M}$ = 1 w 8088);

$\overline{WR}$ – wyjściowy sygnał trójstanowy, sterujący operacjami zapisu do pamięci lub układów we–wy, aktywny w taktach T_2,T_3 i T_w;

$\overline{INTA}$ – wyjściowy trójstanowy sygnał potwierdzenia przerwania, aktywny w taktach T_2, T_3 i T_w;

ALE – wyjściowy sygnał, wskazujący poziomem wysokim (ALE = 1), że na multipleksowanej magistrali AD15...AD0 (AD7...AD0 w procesorze 8088) znajduje się adres;

DT/$\overline{R}$ – wyjściowy sygnał trójstanowy, określający kierunek przesyłania danych przez dwukierunkowe bufory trójstanowe. Przy operacji zapisu sygnał DT/$\overline{R}$ =1, a przy operacji odczytu sygnał DT/$\overline{R}$ = 0.

$\overline{DEN}$ – wyjściowy sygnał trójstanowy, służący do odblokowania dwukierunkowych buforów trójstanowych. Aktywny podczas każdego cyklu odczytu i zapisu pamięci lub układów we–wy oraz podczas cyklu potwierdzenia przerwania;

HOLD – wejściowy sygnał żądania przejścia mikroprocesora do stanu zawieszenia (HOLD) generowany przez inny moduł nadrzędny (żądający dostępu do magistrali). Testowany przez układ sterowania mikroprocesora w takcie T_3. Sygnał HOLD nie jest wewnętrznie synchronizowany;

HLDA – wyjściowy sygnał, potwierdzający przejście mikroprocesora do stanu zawieszenia. Magistrala mikroprocesora przechodzi do stanu dużej impedancji od połowy taktu T_4 (lub T_i).

Wyprowadzenia dla *trybu maksymalnego*

$\overline{S2}...\overline{S0}$ – wyjściowe trójstanowe sygnały statusu mikroprocesora, na podstawie których sterownik magistrali 8288 dekoduje cykl magistrali i określa odpowiadające mu sygnały sterujące (tabl. 2.7). Status określający cykl magistrali pojawia się w taktach T_4, T_1 i T_2. W taktach T_3 lub T_w (dla READY = 1) pojawia się tzw. *status pasywny* $\overline{S2}\ \overline{S1}\ \overline{S0}$ = 111, wskazujący zakończenie cyklu magistrali;

Tablica 2.7

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	Rodzaj cyklu magistrali
0	0	0	Cykl potwierdzenia przerwania
0	0	1	Cykl odczytu układu wejściowego
0	1	0	Cykl zapisu układu wyjściowego
0	1	1	Cykl potwierdzenia rozkazu HLT
1	0	0	Cykl pobrania kodu rozkazu
1	0	1	Cykl odczytu pamięci
1	1	0	Cykl zapisu pamięci
1	1	1	Stan „pasywny" statusu

$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$ (ang. *request grant*) – dwukierunkowe linie żądania dostępu do magistrali i potwierdzenia przyznania magistrali innym modułom nadrzędnym (np. koprocesorowi 8087 lub mikroprocesorowi we–wy 8089). Linia $\overline{RQ}/\overline{GT0}$ ma wyższy priorytet niż linia $\overline{RQ}/\overline{GT1}$;

$\overline{LOCK}$ – wyjściowy sygnał trójstanowy. $\overline{LOCK}$ = 0 blokuje dostęp do magistrali innym modułom nadrzędnym. Sygnał $\overline{LOCK}$ przechodzi do stanu 0 po wykonaniu jednobajtowego rozkazu (przedrostka – ang. *prefix*) LOCK i pozostaje w nim podczas wykonywania następnego rozkazu;

QS0, QS1 – sygnały wyjściowe, wskazujące rodzaj operacji wykonywanej na kolejce rozkazów – (tabl. 2.8).

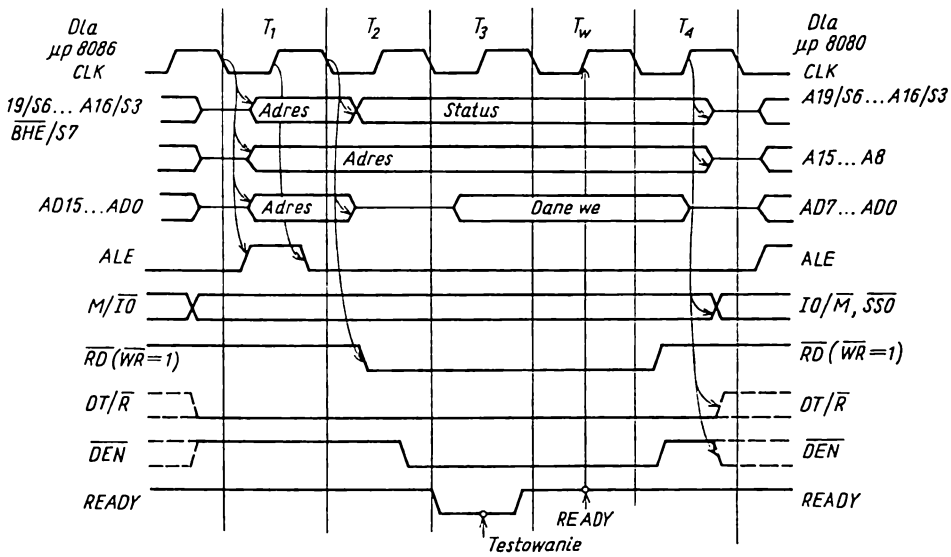
Tablica 2.8

QS0	QS1	Rodzaj operacji
0	0	Brak operacji
0	1	Pobranie pierwszego bajtu rozkazu z kolejki
1	0	Zerowanie kolejki
1	1	Pobranie bajtu rozkazu z kolejki

Cykl magistrali mikroprocesorów 8086/88 składa się z 4 taktów zegarowych $T_1...T_4$ wówczas, gdy mikroprocesor nie przechodzi do stanu oczekiwania lub z 4 taktów

$T_1...T_4$ i nT_w, w przypadku przeciwnym. Podczas realizacji złożonych rozkazów (np. mnożenia) mogą wystąpić okresy, w których nie jest wykonywany żaden cykl magistrali (gdyż EU wykonuje rozkaz, a BIU zapełnił już kolejkę następnymi rozkazami). Takty zegarowe występujące w tych okresach, nazwane *taktami bezczynnymi* (ang. *idle*) magistrali, oznacza się jako T_i.

BIU inicjuje cykl magistrali wówczas, gdy musi zapełniać kolejkę rozkazów lub na żądanie układu EU (w celu odczytania argumentu lub zapisania wyniku). Inicjacja cyklu pobrania kodu rozkazu do kolejki rozkazów następuje gdy:
- 2 (lub więcej) bajty w kolejce są puste, w mikroprocesorze 8086;
- 1 (lub więcej) bajt w kolejce jest pusty, w mikroprocesorze 8088.



Rys. 2.8. Cykl odczytu pamięci lub układu we–wy

Na rysunku 2.8 przedstawiono cykl odczytu pamięci lub układu we–wy w trybie minimalnym mikroprocesora. Cykle pobrania kodu z pamięci, odczytu argumentu z pamięci lub układu we–wy mają jednakowy przebieg, tylko adres fizyczny jest utworzony z innych elementów składowych.

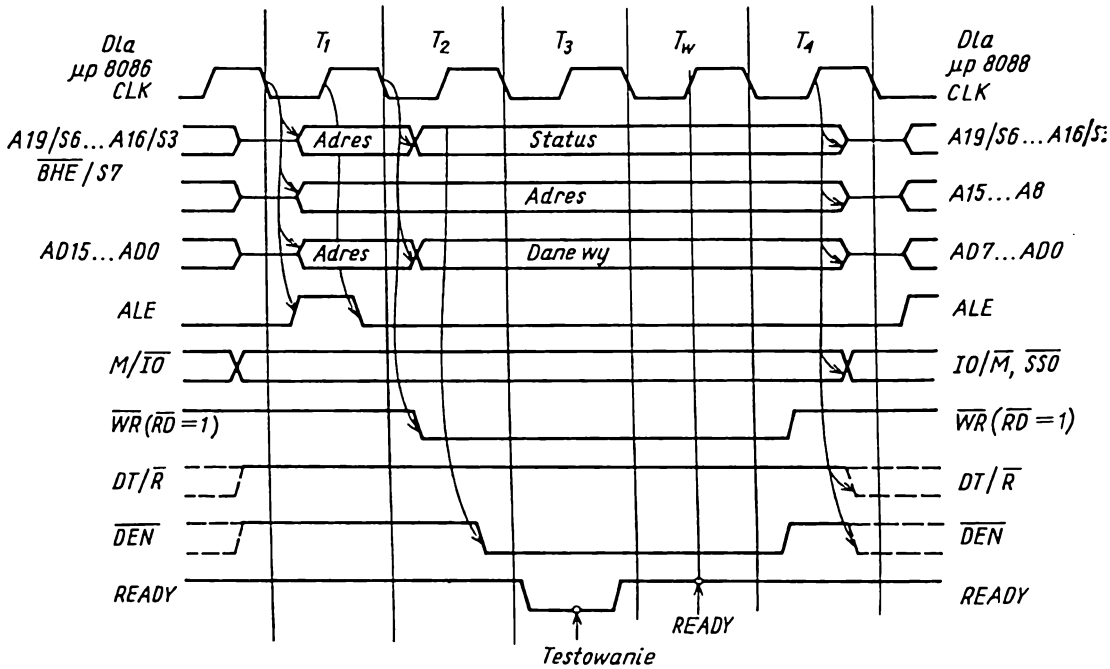
W takcie T_1 mikroprocesor umieszcza na multipleksowanej magistrali adres/dane (adres/status) 20-bitowy adres i generuje sygnał ALE = 1.

W takcie T_2 na liniach A19/S6...A16/S3, w procesorze 8086 również na linii $\overline{BHE}$/S7 pojawia się status, a linie AD15...AD0 w procesorze 8086 (AD7...AD0 w 8088) przechodzą do stanu dużej impedancji. W mikroprocesorze 8088 na liniach A15...A8 do połowy taktu T_4 jest utrzymywany adres (w cyklu potwierdzenia przerwania maskowalnego linie te są w stanie dużej impedancji).

W takcie T_3 na liniach AD15...AD0 (AD7...AD0 w procesorze 8088) powinny pojawić się dane z adresowanego źródła. Przy dużych czasach dostępu źródła mikro-

procesor może wydłużyć cykl odczytu o dodatkowe takty oczekiwania T_w, wtrącane między takty T_3 i T_4 (patrz rys. 2.8 i 2.9). W takcie T_2 na liniach adres/status mikroprocesor umieszcza status, który będzie utrzymywany do połowy taktu T_4.

Sygnał odczytu $\overline{RD}$ jest aktywny w taktach T_2, T_3 (i ewentualnie w taktach T_w). Podobny przebieg ma sygnał $\overline{DEN}$, z tym, że jest on uaktywniany o pół taktu później. Odpowiednie watrości sygnałów wyboru źródła danych (M/$\overline{IO}$ lub IO/$\overline{M}$) oraz kierunku przepływu danych (DT/$\overline{R}$) są ustawiane pod koniec poprzedniego cyklu magistrali i utrzymywane do połowy taktu T_4.



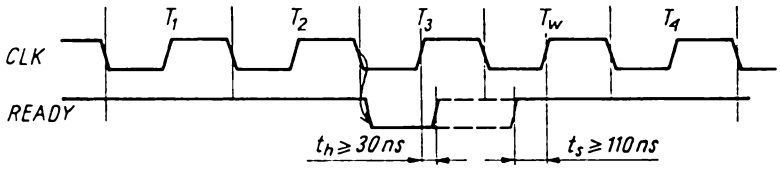
Rys. 2.9. Cykl zapisu pamięci lub układu we–wy

Przebieg *cyklu zapisu* do pamięci lub układów we–wy przedstawiono na rys. 2.9. Jest on bardzo podobny do *cyklu odczytu*, z tym, że mikroprocesor po zdjęciu adresu z multipleksowanych linii adres/dane ustawia na nich dane wyjściowe, które będą utrzymywane do połowy taktu T_4, i generuje sygnał zapisu $\overline{WR}$ = 0 (zamiast sygnału $\overline{RD}$). Sygnał DT/$\overline{R}$ przechodzi w tym cyklu do stanu 1.

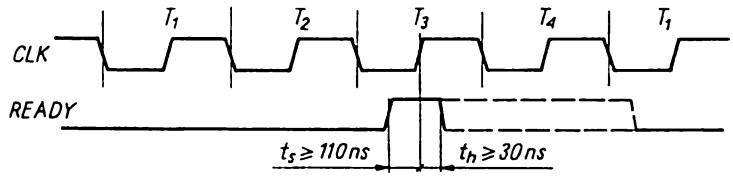
Sygnał READY umożliwia wydłużenie cyklu zapisu lub odczytu, w celu dopasowania szybkości pracy mikroprocesora do szybkości pracy wolnych pamięci lub układów we–wy. Jest on również używany w systemach wieloprocesorowych do wydłużania cyklu magistrali mikroprocesora w przypadku żądania przez niego dostępu do magistrali systemowej w chwili, gdy jest ona zajęta przez inny moduł nadrzędny. Sygnał READY jest doprowadzany do mikroprocesora przez układ generatora impulsów zegarowych (patrz rys. 2.16), w którym następuje jego synchronizacja opadającym zboczem impulsu zegarowego CLK.

Stosuje się dwa sposoby wprowadzania mikroprocesorów 8086/88 w *stan oczekiwania*. Pierwszy z nich (stosowany w niezbyt złożonych systemach mikrokompu-

terowych) polega na tym, że zwykle sygnał READY = 1 (mikrokomputer normalnie gotowy) i dopiero przy odwołaniu do pamięci lub układów we–wy, nie spełniających wymagań czasowych narzuconych przez cykl magistrali mikroprocesora, sygnał READY przechodzi do stanu 0 na wymagany czas – rys. 2.10. W rozbudowanych systemach mikrokomputerowych częściej przyjmuje się, że sygnał READY = 0 (mikrokomputer normalnie niegotowy) i dopiero wybrany układ pamięci lub układy we–wy powodują zmianę stanu sygnału READY z 0 na 1 (rys. 2.11). W obu przypadkach są możliwe zarówno cykle zawierające takty T_w, jak i cykle bez tych taktów. Podane czasy podtrzymania i wyprzedzenia sygnału READY dotyczą mikroprocesora o f_{max} = 5 MHz.

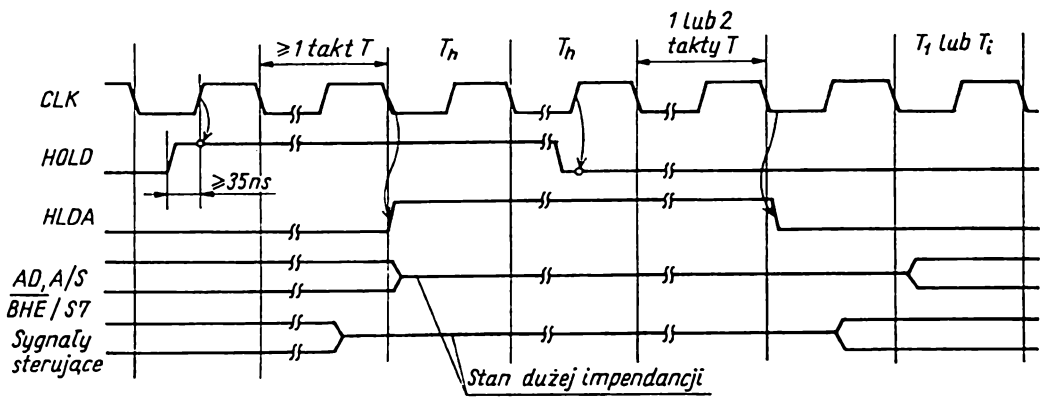


Rys. 2.10. Przykładowy przebieg sygnału READY w mikrokomputerze normalnie gotowym



Rys. 2.11. Przykładowy przebieg sygnału READY w mikrokomputerze normalnie niegotowym

W *stanie oczekiwania* żaden z sygnałów wyjściowych mikroprocesora nie zmienia swojego stanu.



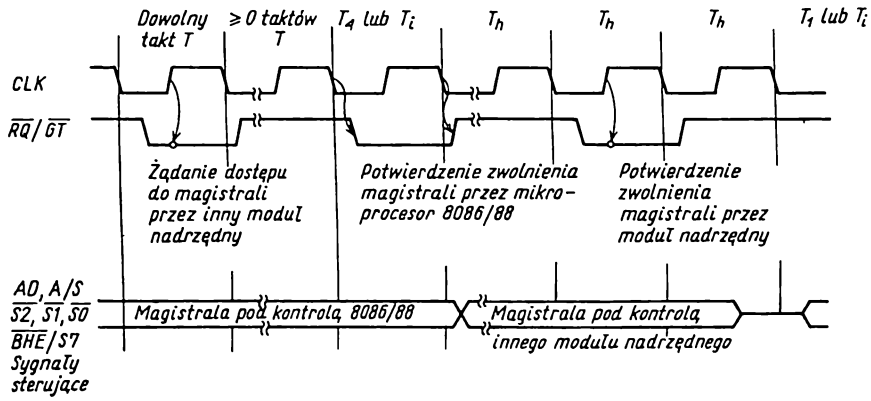
Rys. 2.12. Stany sygnałów przy przejściu mikroprocesora do stanu zawieszenia

Wejście do *stanu zawieszenia* HOLD mikroprocesorów 8086/88, w trybie minimalnym, jest powodowane sygnałem HOLD = 1. Stan sygnału HOLD jest tes-

towany narastającym zboczem impulsu CLK w takcie T_3, każdego cyklu magistrali oraz w każdym takcie T_i i T_h. Mikroprocesor może przejść do stanu zawieszenia i potwierdzić to sygnałem HLDA = 1 po zakończeniu bieżącego cyklu magistrali (jeżeli taki był wykonywany), pod warunkiem, że nie był to pierwszy cykl magistrali cyklu potwierdzenia przerwania lub jeden z cykli magistrali rozkazu poprzedzanego rozkazem *LOCK*.

W stanie zawieszenia linie sterujące, danych i adresowe przechodzą do stanu dużej impedancji. Wyjście ze stanu zawieszenia następuje po zaniku sygnału HOLD. Przejście mikroprocesora do stanu zawieszenia przedstawiono na rys. 2.12.

W trybie maksymalnym żądanie dostępu do magistrali od innego modułu typu "Master" jest przekazywane do mikroprocesora dwukierunkową linią $\overline{RQ}/\overline{GT0}$ (lub $\overline{RQ}/\overline{GT1}$). Tą linią są przekazywane również sygnały potwierdzenia zwolnienia magistrali zarówno z mikroprocesora do układu nadrzędnego, jak i odwrotnie. Zwolnienie magistrali przez mikroprocesor następuje w takich samych przypadkach jak w trybie minimalnym. Typową sekwencję sygnałów żądania dostępu i potwierdzenia zwolnienia magistrali przedstawiono na rys. 2.13.



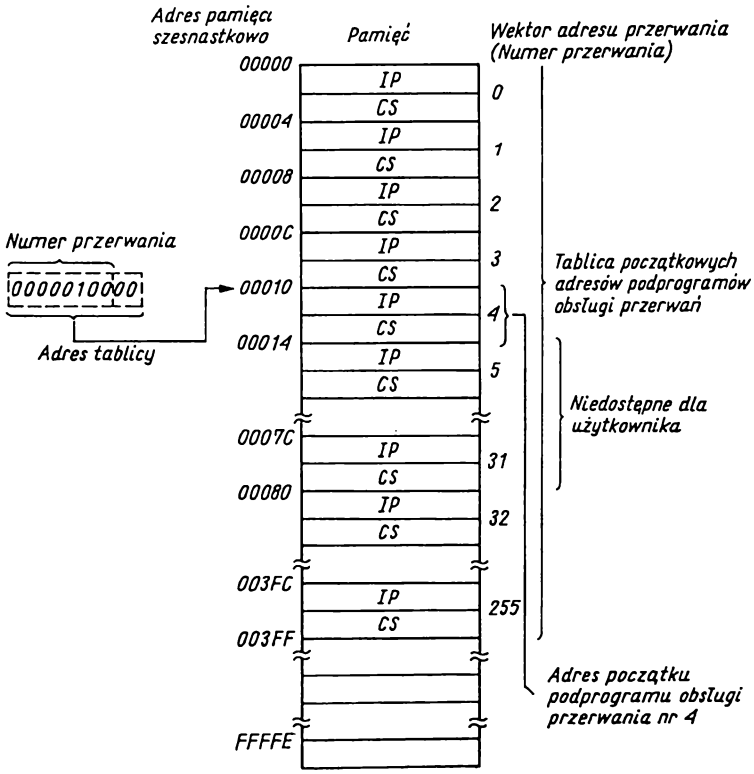
Rys. 2.13. Sekwencja zmian stanu dwukierunkowej linii $\overline{RQ}/\overline{GT}$ zapewniająca zwolnienie i odzyskanie magistrali przez mikroprocesor

Linia $\overline{RQ}/\overline{GT0}$ ma wyższy priorytet niż linia $\overline{RQ}/\overline{GT1}$, z tym, że jeżeli układ nadrzędny połączony linią $\overline{RQ}/\overline{GT1}$ z procesorem uzyskał dostęp do magistrali, to żądanie innego układu nadrzędnego połączonego z procesorem linią $\overline{RQ}/\overline{GT0}$ będzie ignorowane aż do chwili zwolnienia magistrali przez ten układ.

Po wykonaniu rozkazu *HLT* mikroprocesor 8086 (8088) przechodzi do *stanu zatrzymania* i pozostaje w nim do chwili pojawienia się sygnału zerowania RESET lub sygnału żądania przerwania niemaskowalnego NMI, lub maskowalnego INTR (jeśli znacznik IF = 1). Rozkaz *HLT* nie zmienia stanu znaczników w rejestrze F.

Mikroprocesory 8086/88 wyposażono w efektywny system przerwań, reagujący na zdarzenia zewnętrzne i wewnętrzne. Logiczne adresy początków programów obsługi tych zdarzeń znajdują się w tablicy (rys. 2.14) zajmującej komórki od 00000H do 003FFH w jednomegabajtowej przestrzeni adresowej pamięci. Adres logiczny, na

podstawie którego BIU wyznacza adres fizyczny początku danego podprogramu obsługi, zajmuje 4 bajty w tablicy. Dwa pierwsze wskazują położenie pierwszego rozkazu podprogramu obsługi przerwania w segmencie pamięci programu (ich zawartość jest zapisywana do wskaźnika rozkazów IP), a dwa następne – adres początku segmentu pamięci programu (zapisywany do rejestru CS). Tablica zawiera więc 256 logicznych adresów początków podprogramów obsługi przerwań, identyfikowanych przez podanie wektora adresu przerwania (numeru lub poziomu przerwania n = 0...255). Adres fizyczny słowa wskazującego zawartość wskaźnika IP wyznacza się jako $4n$, a adres słowa wskazującego zawartość rejestru CS – jako $4n + 2$.



Rys. 2.14. Wykaz logicznych adresów początków podprogramów obsługi przerwań

Przerwnia wewnętrzne:

INTn – przerwanie programowe, występujące bezpośrednio po zakończeniu wykonywania 2-bajtowego rozkazu, *INTn* umieszczonego w programie. Parametr n wskazuje numer wektora adresu przerwania w tablicy. Rozkaz *INT*3, w przeciwieństwie do pozostałych rozkazów *INTn* ($n \neq 3$), jest 1-bajtowy. Zwykle stosuje się go do

ustawiania punktów kontrolnych (ang. *breakpoint*) w uruchamianych programach;

INTO – przerwanie programowe, występujące bezpośrednio po zakończeniu wykonywania 1-bajtowego rozkazu *INT*0, jeśli znacznik OF = 1 (wystąpiło przepełnienie w operacjach poprzedzających wykonanie rozkazu *INTO*). Przerwaniu temu odpowiada stały adres początku podprogramu obsługi określony przez wektor o numerze $n = 4$;

- przerwanie przy błędzie dzielenia (ang. *divide error*). Jest ono generowane automatycznie w układzie sterowania procesora po zakończeniu wykonywania rozkazu *DIV* (lub *IDIV*), jeżeli wartość wyliczonego ilorazu przekracza dopuszczalny zakres (np. przy próbie dzielenia przez zero). Przerwaniu temu odpowiada stały adres początku podprogramu obsługi określony przez wektor o numerze $n = 0$;
- przerwanie generowane po zakończeniu wykonywania każdego rozkazu, jeśli *znacznik pułapki* TF = 1. Umożliwia ono krokową realizację programów; odpowiada mu stały adres początku podprogramu obsługi okraślany przez wektor o numerze $n = 1$. Gdy znacznik TF = 0, przerwanie jest zablokowane. Ze względu na brak rozkazów umożliwiających bezpośrednią programową zmianę znacznika TF, stosuje się do tego celu rozkazy *PUSH* F, rozkazy modyfikujące i rozkaz *POP* F. Należy jednak podkreślić, że żadne przerwanie nie jest przyjmowane między przedrostkami np. blokady *LOCK* czy powtórzenia – *REP* a następującymi po nich rozkazami, lecz dopiero po tych rozkazach (w przypadku *REP* po każdym powtórzeniu).

Przerwania zewnętrzne

- przerwanie niemaskowalne (nie można go zablokować w programie), wprowadzane do mikroprocesora przez wejście NMI. Jest ono inicjowane narastającym zboczem sygnału NMI. Jego wystąpienie jest zapamiętywane w wewnętrznym układzie przerwań i zawsze akceptowane w mikroprocesorze po zakończeniu wykonywania bieżącego rozkazu. Przerwaniu niemaskowalnemu odpowiada stały wektor adresu przerwania oznaczony numerem $n = 2$;
- przerwanie maskowalne, inicjowane sygnałem INTR = 1 przy znaczniku IF = 1. Przerwanie to można zablokować w programie rozkazem *CLI*, zerującym znacznik zezwolenia na przerwanie IF. Znacznik IF jest również zerowany po przyjęciu przerwania oraz po zerowaniu sygnałem RESET. Odblokowanie wejścia przerwań maskowalnych polega na ustawieniu stanu znacznika IF na 1 rozkazem *STI*.

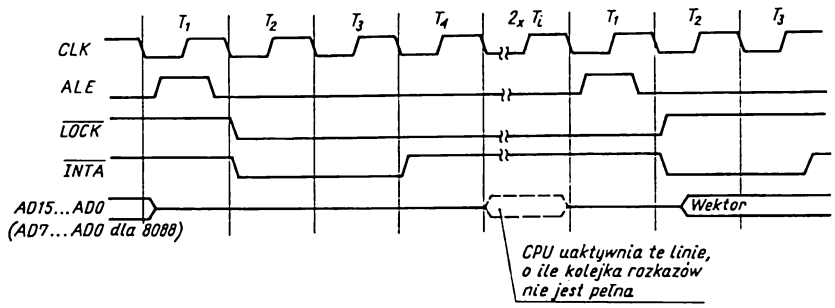
Żądanie przerwania maskowalnego nie jest zapamiętywane w wewnętrznym układzie przerwań. Stan sygnału INTR = 1 powinien więc być utrzymywany do chwili potwierdzenia przyjęcia przerwania przez mikroprocesor. Wejście przerwań masko-

walnych przystosowano do współpracy z układem sterowania przerwań 8259A (p. 3.1). Wektor adresu przerwania jest przekazywany z układu 8259A do mikroprocesora w cyklu potwierdzenia przerwania.

Użytkownik, przy przerwaniach maskowalnych i programowych INTR*n*, powinien stosować wektory adresów przerwań o numerze $n = 32...255$. Stosowanie pozostałych jest możliwe, ale niewskazane, gdyż są one przyporządkowane przerwaniom wewnętrznym i przerwaniu niemaskowalnemu lub zarezerwowane przez producenta dla potrzeb oprogramowania systemowego.

Przyjęcie przerwania wewnętrznego nie wymaga cyklu potwierdzenia przerwania na magistrali. Procesor nie "czeka" na wektor adresu przerwania. Następuje przy tym odłożenie na stosie zawartości rejestru znaczników F, rejestru segmentu programu CS i wskaźnika rozkazów IP; zerowanie znaczników IF i TF oraz załadowanie do rejestru CS słowa, znajdującego się w tablicy logicznych adresów początków podprogramów obsługi przerwań pod adresem $4n + 2$, a do wskaźnika rozkazów IP słowa, znajdującego się pod adresem $4n$ (n – numer przerwania).

Przyjęcie przerwania niemaskowalnego odbywa się identycznie. Po przyjęciu przerwania maskowalnego (przy znaczniku IF = 1), układ BIU inicjuje cykl potwierdzenia przerwania złożony z dwóch cykli magistrali podobnych do cykli odczytu pamięci, z tym, że zamiast impulsu $\overline{RD}$ pojawia się w trybie minimalnym impuls $\overline{INTA}$, a w trybie maksymalnym – status $\overline{S2}\ \overline{S1}\ \overline{S0} = 000$, na podstawie którego sterownik magistrali wygeneruje sygnały $\overline{INTA}$. Pierwszy sygnał $\overline{INTA} = 0$ sygnalizuje układowi 8259A, że przerwanie zostało przyjęte (rys. 2.15). Podczas drugiego sygnału $\overline{INTA} = 0$ sterownik przerwań przesyła liniami danych wektor adresu przerwania n, na podstawie którego BIU wyznacza adres fizyczny początku podprogramu obsługi. Kolejność odkładania stanu rejestrów F, CS i IP oraz ładowania rejestrów CS i IP nowymi wartościami jest identyczna z omówioną przy przerwaniach wewnętrznych. Podobnie zeruje się również znaczniki IF i TF.



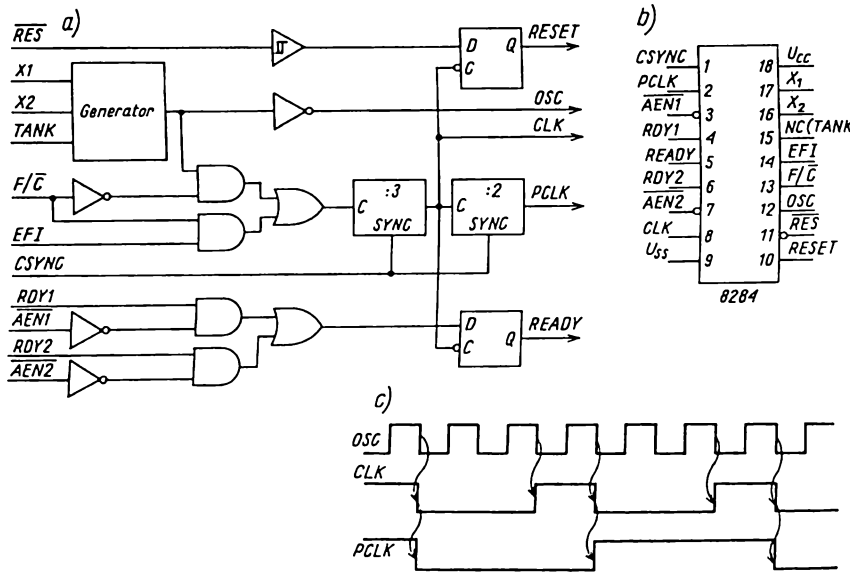
Rys. 2.15. Cykl potwierdzenia przerwania

Należy dodać, że podczas cyklu potwierdzenia przerwania maskowalnego, sygnały HOLD (w trybie minimalnym) i $\overline{RQ}/\overline{GT}$ (w trybie maksymalnym) są ignorowane. W trybie maksymalnym jest ponadto generowany sygnał $\overline{LOCK} = 0$, blokujący innym modułom nadrzędnym dostęp do magistrali.

Każdy podprogram obsługi przerwania kończy się rozkazem powrotu *IRET*. Rozkaz ten powoduje odtwarzanie zawartości rejestrów IP, CS i F.

Przy jednoczesnym wystąpieniu kilku różnych przerwań (zewnętrznych i wewnętrznych) maskowalne przerwanie INTR, o najniższym priorytecie, będzie obsłużone jako ostatnie, jeśli będzie odblokowane (po zakończeniu obsługi przerwań o wyższym priorytecie). Przy jednoczesnym wystąpieniu przerwania dla pracy krokowej i przerwania NMI lub przerwania programowego, wyższy priorytet ma przerwanie dla pracy krokowej, a przy jednoczesnym wystąpieniu przerwania NMI i przerwania programowego, wyższy priorytet ma przerwanie NMI. Przy jednoczesnym wystąpieniu tych trzech przerwań kolejność ich obsługi będzie następująca: najpierw program obsługi przerwania NMI, potem przerwania programowego (oba bez pracy krokowej), a następnie program obsługi pracy krokowej (po rozkazie następującym za rozkazem powodującym przerwanie programowe).

Jednostka centralna z mikroprocesorem pracującym w trybie minimalnym musi zawierać układ generowania impulsów zegarowych (8284), rejestry z wyjściami trójstanowymi (przechowujące adres podczas cyklu magistrali) i w razie potrzeby dwukierunkowe trójstanowe bufory magistrali danych. *Układ zegara* 8284 generuje impulsy CLK synchronizujące pracę wewnętrznych układów mikroprocesora i PCLK synchronizujące układy we-wy oraz zapewnia synchronizację sygnałów READY i RESET przed wprowadzeniem ich do mikroprocesora (rys. 2.16). Na wyjściu OSC pojawiają się impulsy o częstotliwości równej częstotliwości rezonatora kwarcowego.



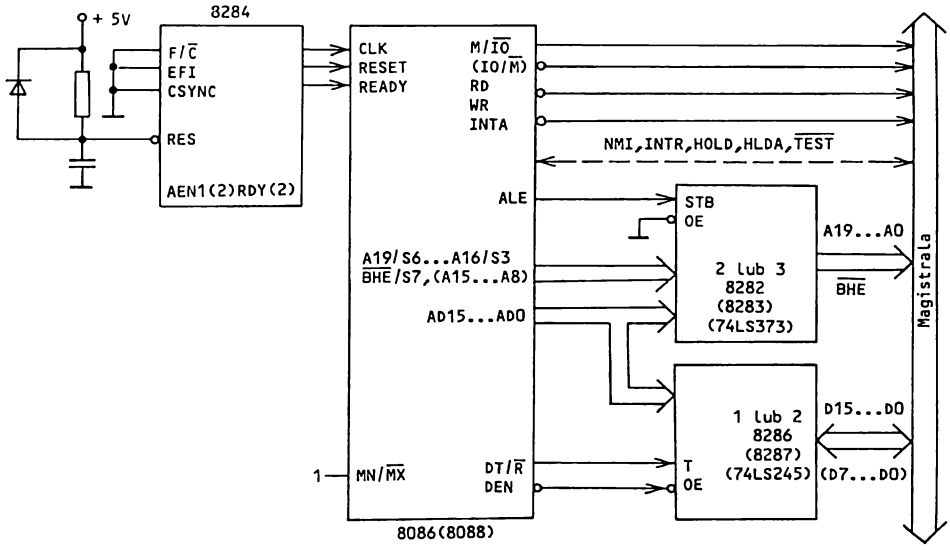
Rys. 2.16. Układ zegara 8284: a) schemat blokowy; b) rozmieszczenie wyprowadzeń; c) przebiegi czasowe

Znaczenie pozostałych wyprowadzeń

RES – wejściowy sygnał zerujący (wejście typu przerzutnik Schmitta, łączone zwykle z obwodem RC generującym sygnał zerujący o wymaganym czasie trwania);

RESET – wyjściowy sygnał zerujący, doprowadzany do mikroprocesora i innych współpracujących z nim układów;
EFI – wejściowy sygnał zewnętrznych impulsów zegarowych;
CSYNC – wejściowy sygnał synchronizujący pracę kilku układów 8284. Przy CSYNC = 1 wewnętrzne liczniki (dzielniki) są wyzerowane, a przy CSYNC = 0 liczniki rozpoczynają pracę od zmiany stanu sygnałów OSC, CLK i PCLK z 1 na 0. Przy stosowaniu zewnętrznych impulsów zegarowych EFI, sygnał CSYNC powinien być synchronizowany sygnałem EFI;
F/$\overline{C}$ – wejściowy sygnał wybierający źródło impulsów o częstotliwości podstawowej. Przy F/$\overline{C}$ = 1 jest wybrane źródło zewnętrzne (EFI), a przy F/$\overline{C}$ = 0 – wewnętrzne (OSC);
RDY1, RDY2 – wejściowe sygnały gotowości, generowane przez układy we–wy;
READY – wyjściowy sygnał gotowości, kierowany do mikroprocesora;
AEN1, AEN2 – wejściowe sygnały blokujące sygnały gotowości;
x_1, x_2 – wejścia, do których dołącza się rezonator kwarcowy.

Moduł jednostki centralnej z mikroprocesorem 8086 lub 8088 pracującym w trybie minimalnym przedstawiono na rys. 2.17.

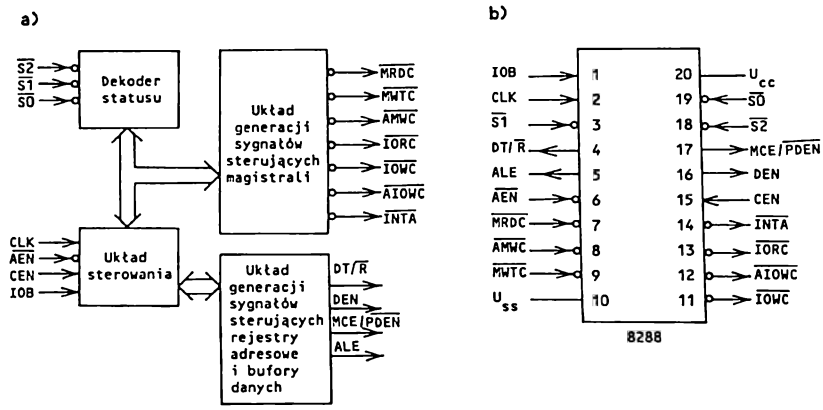


Rys. 2.17. Jednostka centralna z mikroprocesorem 8086 (8088)

W jednostce centralnej z mikroprocesorem 8088 sygnały adresowe A15...A8 są utrzymywane przez cały czas trwania cyklu magistrali (patrz rys. 2.8 i 2.9). Nie muszą więc być wpisywane do zewnętrznego rejestru.

Jednostka centralna z mikroprocesorem pracującym w trybie maksymalnym umożliwia konstruowanie mikrokomputerów jedno i wieloprocesorowych z magistralą systemową zgodną z Multibus i ewentualnie z magistralami lokalnymi. W tym przypadku jednostka centralna musi zawierać dodatkowo *sterownik magistrali* 8288 oraz w przypadku systemu wielomikroprocesorowego – *układ arbitra magistrali* 8289.

Sterownik magistrali (rys. 2.18), na podstawie słowa statusu mikroprocesora 8086/88 ($\overline{S2}$ $\overline{S1}$ $\overline{S0}$ – patrz tabl. 2.7), dekoduje rodzaj cyklu magistrali i określa niezbędne dla niego sygnały sterujące. Część sygnałów wejściowych ($\overline{S2}$, $\overline{S1}$, $\overline{S0}$, CLK) i wyjściowych (DT/$\overline{R}$, $\overline{INTA}$, ALE, DEN) sterownika magistrali spełnia takie same funkcje, jak ich odpowiedniki w mikroprocesorach 8086 i 8088. Znaczenie pozostałych sygnałów jest następujące:



Rys. 2.18. Sterownik magistrali 8288: a) schemat blokowy; b) rozmieszczenie wyprowadzeń

$\overline{MWTC}$ – wyjściowy sygnał zapisu danych do pamięci;

$\overline{AMWC}$ – wyjściowy, wydłużony sygnał zapisu danych do pamięci. Umożliwia on zapisanie danych do pamięci o dłuższym czasie dostępu, bez wprowadzania dodatkowego taktu T_w;

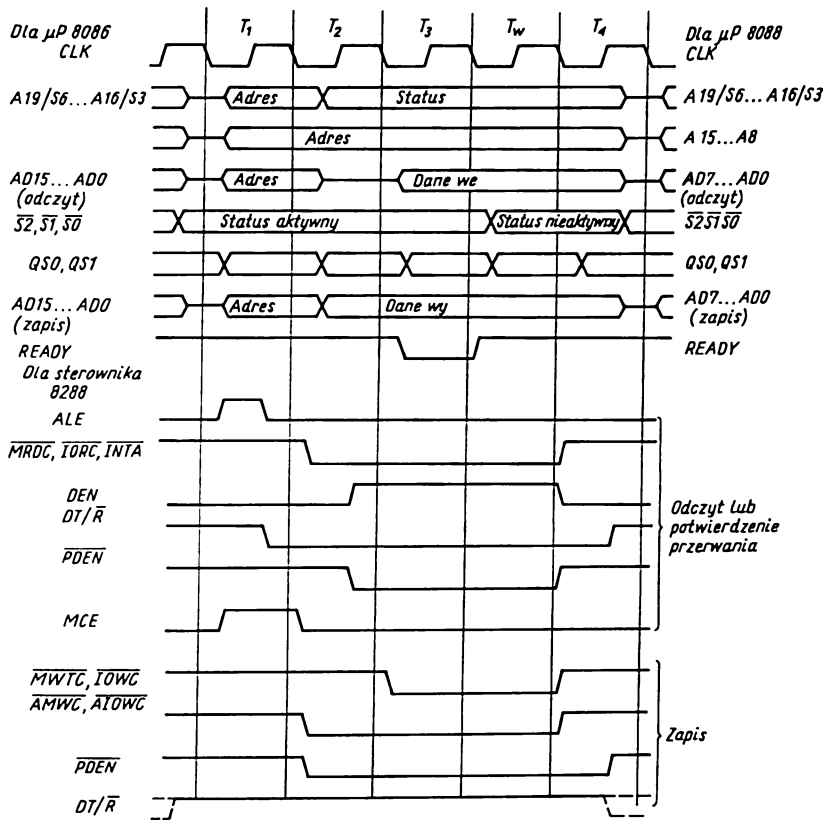
$\overline{MRDC}$ – wyjściowy sygnał odczytu z pamięci;

$\overline{IOWC}$ – wyjściowy sygnał zapisu danych do układów we–wy;

$\overline{AIOWC}$ – wyjściowy, wydłużony sygnał zapisu danych do układów we–wy. Umożliwia on zapisanie danych do układów we–wy o dłuższym czasie dostępu, bez wprowadzania dodatkowego taktu T_w;

$\overline{IORC}$ – wyjściowy sygnał odczytu danych z układów we–wy;

IOB – wejściowy sygnał wyboru trybu pracy sterownika. Przy IOB = 1 będzie wybrany tryb pracy z lokalną magistralą we–wy. W tym, trybie wszystkie odwołania do pamięci odbywają się magistralą systemową Multibus, a odwołania do układów we–wy – magistralą lokalną. Przy IOB = 0 będzie wybrany tryb pracy z systemową magistralą Multibus;



Rys. 2.19. Cykl zapisu i odczytu pamięci i układów we–wy w mikroprocesorze pracującym w trybie maksymalnym

$\overline{\text{AEN}}$ – wejściowy sygnał odblokowujący ($\overline{\text{AEN}}$ = 0) wyjściowe bufory trójstanowe sygnałów sterujących cyklu magistrali. Przy $\overline{\text{AEN}}$ = 1 sygnały sterujące są w stanie dużej impedancji, z tym, że nie dotyczy to sygnałów sterujących cyklu we–wy jeżeli sterownik współpracuje z lokalną magistralą we–wy (sygnał IOB = 1);

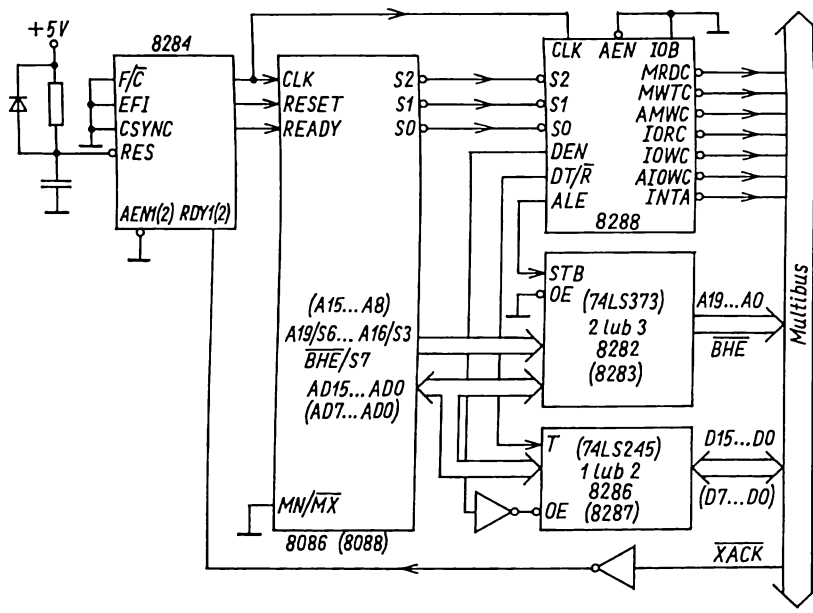
MCE/$\overline{\text{PDEN}}$ – wyjściowy sygnał spełniający różne funkcje zależnie od stanu sygnału wejściowego IOB. Przy IOB = 0 (sterownik współpracuje z magistralą systemową) sygnał MCE = 1 pojawia się podczas cyklu potwierdzenia przerwania. Umożliwia on przesłanie adresu (liniami CAS0...CAS2) ze sterownika przerwań 8259A MASTER (dołączonego do magistrali lokalnej), do sterownika przerwań SLAVE (dołączonego do magistrali sys-

temowej), który wygenerował sygnał żądania przerwania. Przy IOB = 1 sygnał $\overline{PDEN}$ = 0 umożliwia odblokowanie dwukierunkowych buforów magistrali danych podczas cykli we–wy. Spełnia on więc taką samą funkcję, jak sygnał DEN = 1 przy współpracy sterownika z magistralą systemową;

CEN – wejściowy sygnał uaktywniający sterownik magistrali (CEN = 1). Przy CEN = 0 wszystkie sygnały sterujące cyklu magistrali oraz sygnały DEN i $\overline{PDEN}$ przechodzą do stanu nieaktywnego. Umożliwia on w mikroprocesorach 8086/88 uaktywnienie jednego z dwóch sterowników magistrali, z których jeden generuje sygnały sterujące pamięci i układy we–wy dołączone do magistrali systemowej, a drugi – sygnały sterujące pamięci i układy we–wy dołączone do magistrali lokalnej.

Przebiegi czasowe sygnałów, generowanych przez mikroprocesory 8086/88 (pracujące w trybie maksymalnym) i przez sterownik magistrali 8288 w cyklach odczytu i zapisu, przedstawiono na rys. 2.19.

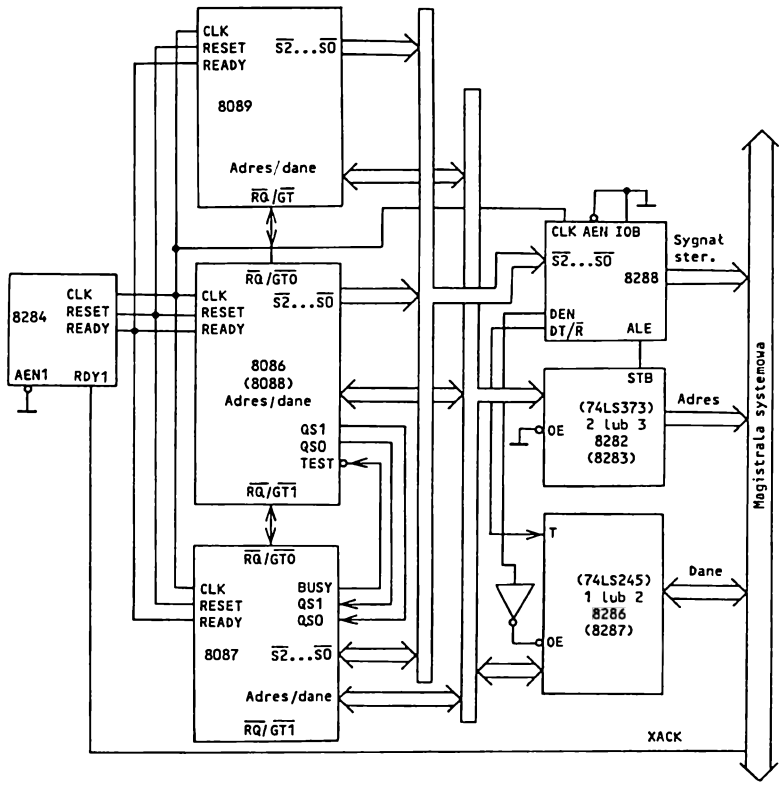
Przykładowe rozwiązanie jednostki centralnej z mikroprocesorem 8086 (8088), pracującym w trybie maksymalnym z magistralą Multibus, do której dołączono pamięci i układy we–wy, przedstawiono na rys. 2.20.



Rys. 2.20. Jednostka centralna z mikroprocesorem pracującym w trybie maksymalnym

Wykorzystując linie $\overline{RQ/GT0}$ i $\overline{RQ/GT1}$, można rozszerzyć jednostkę centralną o *specjalizowany mikroprocesor we–wy* 8089 i *koprocesor arytmetyczny* 8087.

Mikroprocesor we–wy 8089 może wykonywać programy oparte na własnej liście rozkazów i może realizować szybkie przesłania danych typu pamięć ↔ pamięć, we–wy ↔ pamięć, we–wy ↔ we–wy w dwu kanałach DMA.



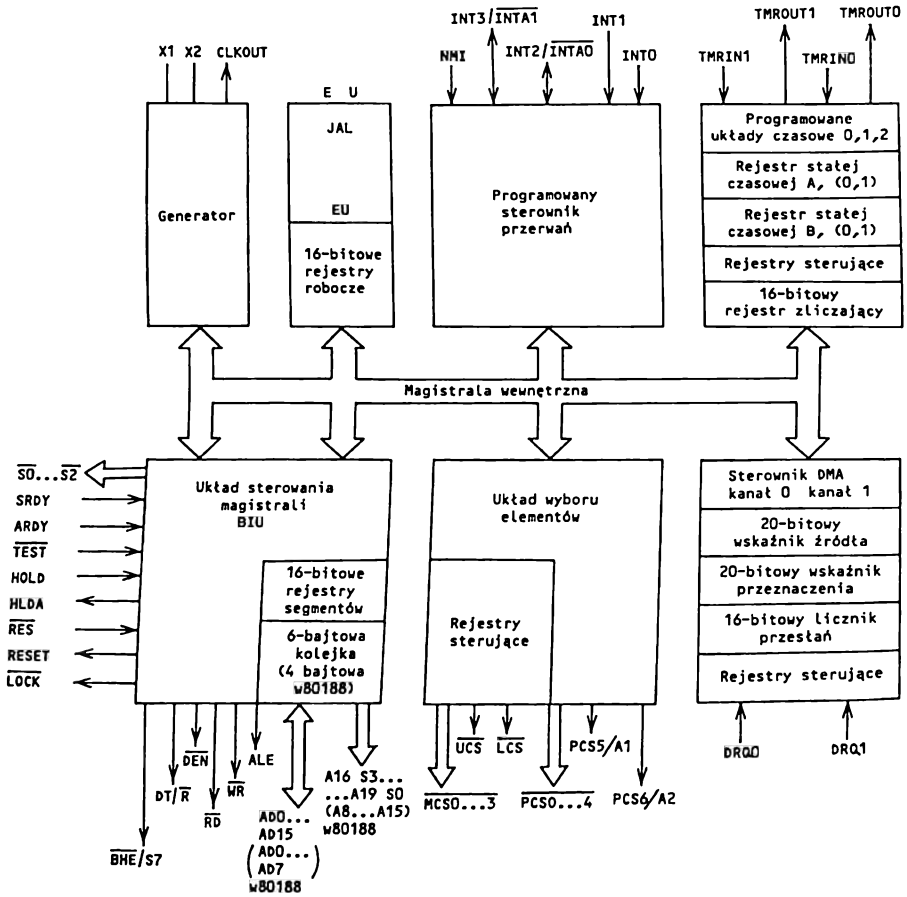
Rys. 2.21. Architektura jednostki centralnej rozszerzonej o specjalizowany procesor we–wy i koprocesor arytmetyczny

Koprocesor arytmetyczny 8087 musi współpracować z mikroprocesorem głównym, dla którego był opracowany (8086/88/186/188). Umożliwia on rozszerzenie listy rozkazów mikroprocesora głównego o rozkazy złożonych operacji arytmetycznych (patrz p. 2.4). Architekturę jednostki centralnej z mikroprocesorem we–wy i koprocesorem przedstawiono na rys. 2.21. Sygnał wyjściowy BUSY przechodzi do stanu 1 od chwili zdekodowania rozkazu (*ESC*) przez koprocesor do chwili zakończenia jego wykonywania. Umożliwia on synchronizację pracy mikroprocesora głównego przez koprocesor (patrz opis sygnału wejsciowego TEST mikroprocesora 8086/88) w sytuacjach, w których w mikroprocesorze istnieje potrzeba skorzystania z wyniku operacji wykonywanej równolegle przez koprocesor.

2.2. Mikroprocesory 80186 i 80188

Mikroprocesory 16-bitowe 80186/188 charakteryzuje duży stopień scalenia, zapewniający umieszczenie w jednej strukturze jednostki centralnej (CPU), o parametrach odpowiadających mikroprocesorom 8086/88, dwóch szybkich kanałów bezpośredniego dostępu do pamięci (DMA); programowalnego sterownika przerwań; trzech programowanych 16-bitowych układów czasowych; programowalnego układu wyboru pamięci i urządzeń zewnętrznych (układów we-wy); programowalnego generatora stanów WAIT; sterownika magistrali lokalnej i generatora impulsów zegarowych.

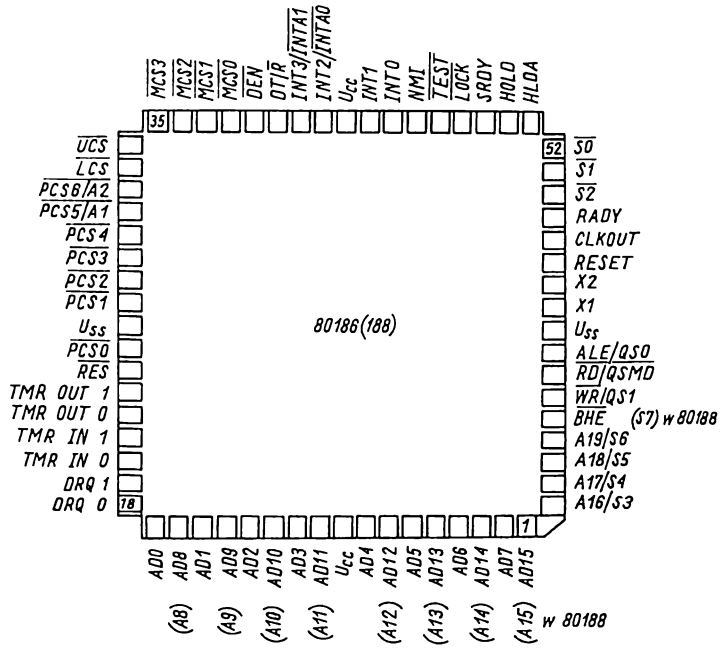
Z punktu widzenia użytkownika są one zgodne ze swymi poprzednikami 8086 i 8088, przy czym ich lista rozkazów zawiera 10 nowych typów rozkazów. Identyczne są również w nich organizacja pamięci, sposób wyznaczania adresu fizycznego, jak również przestrzeń adresowa równa 1 MB.



Rys. 2.22. Struktura wewnętrzna mikroprocesorów 80186 (80188)

Mikroprocesor 80188 ma 16-bitową architekturę wewnętrzną i 8-bitową zewnętrzną magistralę danych (tak, jak procesor 8088). Blokową strukturę wewnętrzną tych mikroprocesorów przedstawiono na rys. 2.22, a rozmieszczenie ich wyprowadzeń na rys. 2.23. Część spośród 68 wyprowadzeń spełnia funkcje identyczne z funkcjami odpowiednich wyprowadzeń mikroprocesorów 8086/88 i charakteryzuje się identycznymi przebiegami czasowymi, występujących na nich sygnałów.

Sygnały $\overline{\text{TEST}}$, NMI, A19/S6...A16/S3, AD15...AD0 (AD7...AD0 i A15...A8 w mikroprocesorze 80188), BHE/S7 (S7 w mikroprocesorze 80188), $\overline{\text{LOCK}}$, $\overline{\text{S2}}$ $\overline{\text{S1}}$ $\overline{\text{S0}}$ mają przebiegi identyczne ze swoimi odpowiednikami w mikroprocesorach 8086/88 w trybie maksymalnym. Dzięki temu w systemach mikrokomputerowych, zawierających mikroprocesor 80186 lub 80188, można stosować sterowniki magistrali 8288, arbitry magistrali 8289, a także koprocesory 8087. Sygnały DT/$\overline{\text{R}}$, $\overline{\text{DEN}}$, HOLD i HLDA spełniają takie same funkcje, jak ich odpowiedniki w mikroprocesorach 8086/88 w trybie minimalnym.



Rys. 2.23. Rozmieszczenie wyprowadzeń mikroprocesorów 80186 (80188)

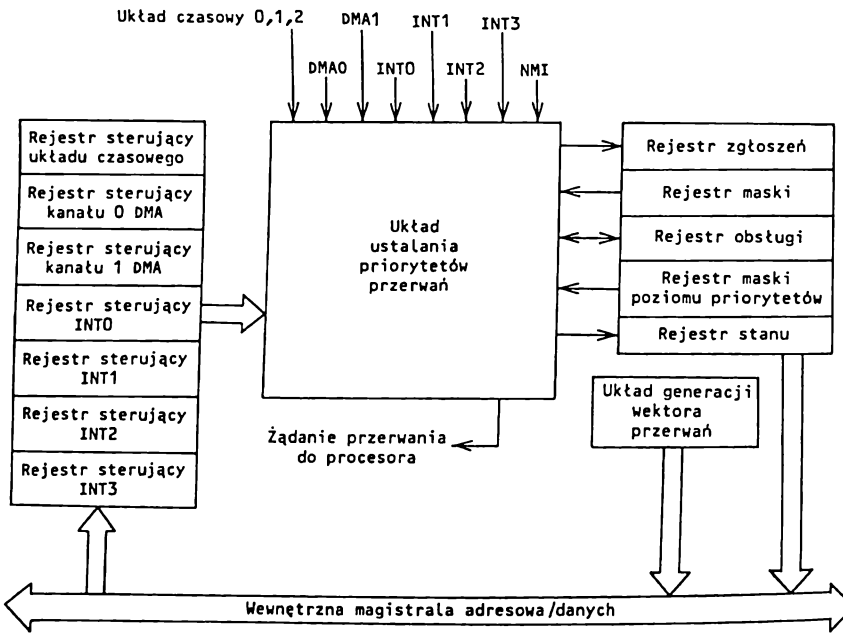
Na wyprowadzeniach ALE/QS0 i $\overline{\text{WR}}$/QS1 pojawiają się sygnały ALE i $\overline{\text{WR}}$ (o takim samym znaczeniu, jak ich odpowiedniki w procesorach 8086/88 w trybie minimalnym) lub sygnały QS0, QS1 (o takim samym znaczeniu, jak ich odpowiedniki w procesorach 8086/88 w trybie maksymalnym). Wybór funkcji tych wyprowadzeń odbywa się za pomocą wyprowadzenia $\overline{\text{RD}}$/$\overline{\text{QSMD}}$, na którym normalnie jest generowany sygnał odczytu pamięci lub układu we-wy. Podczas zerowania układ sterowania

mikroprocesora testuje stan tego wyprowadzenia. Przy niskim stanie (wyprowadzenie $\overline{RD}/\overline{QSMD}$ dołączone do masy) na wyprowadzeniach ALE/QS0 i $\overline{WR}$/QS1 pojawia się stan kolejki.

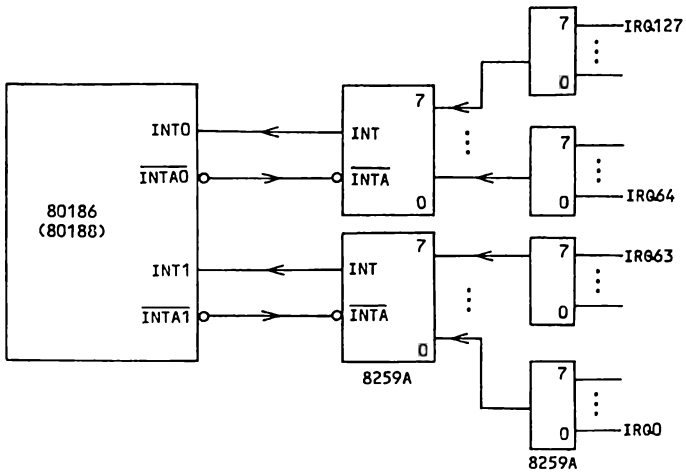
Sygnał wejściowy SRDY (ang. *synchronous ready*) musi być synchronizowany na zewnątrz mikroprocesorów 80186/188. Jest on odpowiednikiem wejścia READY w mikroprocesorach 8086/88. Mikroprocesory 80186/188 zawierają wewnętrzny układ generatora impulsów zegarowych o strukturze i przeznaczeniu odpowiadającym układowi 8284. Do układu tego doprowadza się sygnały ARDY (ang. *asynchronous ready*), $\overline{RES}$, x_1 i x_2, a wyprowadza się sygnały RESET i CLKOUT. Mają one takie przeznaczenie, jak ich odpowiedniki w układzie 8284. Pozostałe wyprowadzenia (z wyjątkiem zasilania) nie mają swoich odpowiedników w mikroprocesorach 8086/88.

Sygnały INT0, INT1 umożliwiają przesłanie przerwań maskowalnych z urządzeń zewnętrznych do wewnętrznego sterownika przerwań o strukturze zbliżonej do struktury sterownika 8259A – rys. 2.24.

Wyprowadzenia INT2/$\overline{INTA0}$ i INT3/$\overline{INTA1}$ można ustawiać w programie jako wejścia (wtedy są to kolejne dwa sygnały żądania przerwań maskowalnych) lub jako wyjścia (wtedy są to sygnały potwierdzenia przyjęcia przerwania maskowalnego odpowiednio $\overline{INTA0}$ i $\overline{INTA1}$). Ta druga opcja umożliwia dołączenie do mikroprocesora zewnętrznych sterowników 8259A w strukturze kaskadowej – rys. 2.25 (każda para INT0, $\overline{INTA0}$ i INT1, $\overline{INTA1}$ umożliwia zwiększenie liczby żądań przerwań maskowalnych do 64).

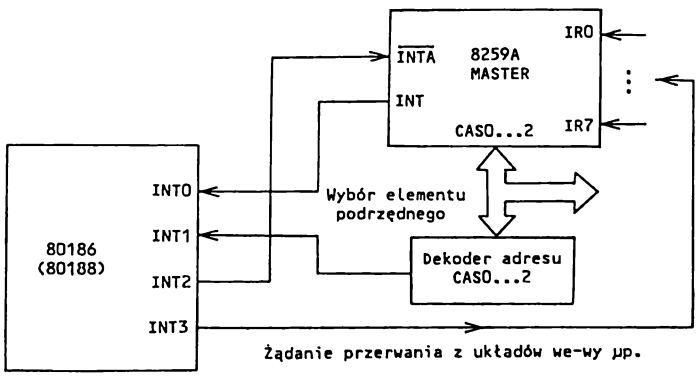


Rys. 2.24. Wewnętrzny sterownik przerwań



Rys. 2.25. Wewnętrzny sterownik przerwań w strukturze kaskadowej

Wewnętrzny sterownik przerwań może pracować jako układ:
- **pojedynczy**, zapewniający obsługę czterech żądań przerwań zewnętrznych;
- **kaskadowy** (każdą parą INT0, $\overline{\text{INTA0}}$ i INT1, $\overline{\text{INTA1}}$ może obsługiwać do 64 przerwań w *trybie pełnego zagnieżdżania*);
- **kaskadowy specjalny**, w którym sterownik MASTER pracuje w *trybie specjalnego pełnego zagnieżdżania* (tryb ten umożliwia przerwanie procedury obsługi przerwania sterownika podrzędnego przez przerwanie o wyższym priorytecie, pojawiające się w tym sterowniku).

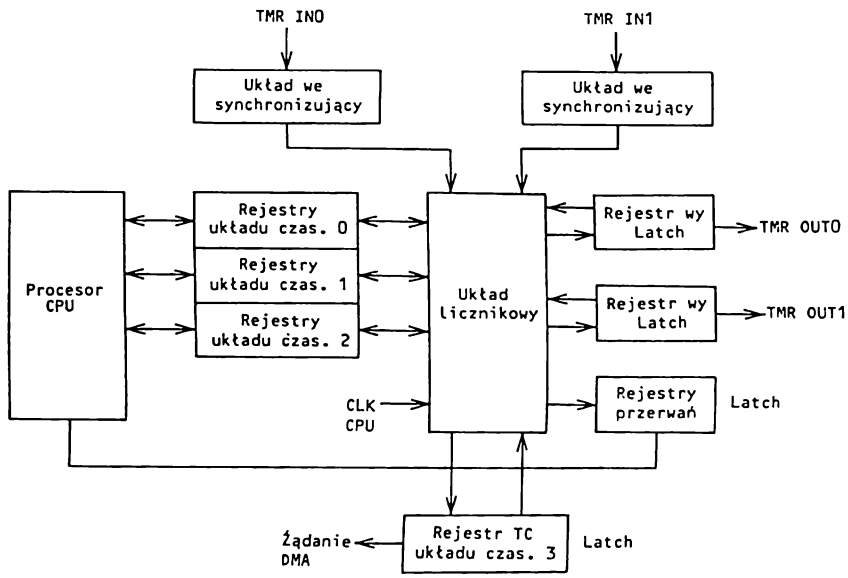


Rys. 2.26. Sterownik przerwań w trybie IRMX 86

Pozostałe wejścia wewnętrznego sterownika przerwań wykorzystuje się do przyjmowania sprzętowych przerwań wewnętrznych (z trzech układów czasowych i dwóch kanałów DMA).

Wewnętrzny sterownik przerwań można również ustawić w programie w *tryb IRMX 86*, w którym stosuje się jeden zwnętrzny sterownik MASTER i wiele sterowników SLAVE (wewnętrzny sterownik przerwań pełni w tym przypadku również funkcję sterownika SLAVE – linie INT1, INT3) – rys. 2.26. W tym przypadku linia INT0 spełnia funkcję wejścia żądania przerwania dla zewnętrznego sterownika MASTER, a linia INT2 – funkcję sygnału INTA. Linią INT3 wewnętrzny sterownik przerwań zgłasza przerwanie sterownikowi MASTER, a linią INT1 sterownik wewnętrzny uzyskuje informację o tym, że jego przerwanie przyjęto i ma on umieścić na magistrali wektor adresu tego przerwania. Ten tryb pracy zapewnia w mikroprocesorach 80186/188 strukturę przerwań zgodną z wymaganiami systemu operacyjnego IRMX 86.

Mikroprocesory 80186/188 zawierają 3 *wewnętrzne programowane układy czasowe* o numerach 0,1 i 2 – rys. 2.27.

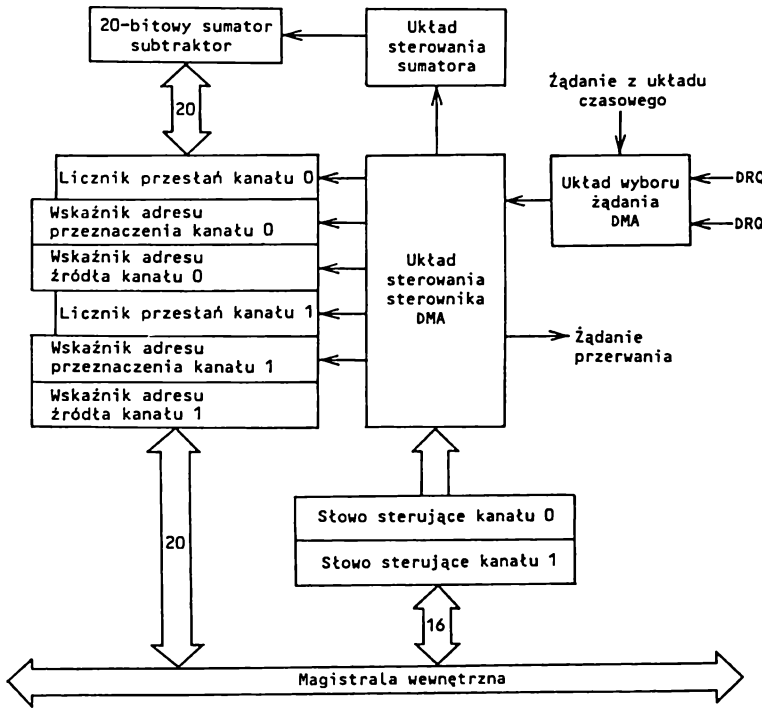


Rys. 2.27. Wewnętrzny układ czasowy

Układy czasowe o numerach 0 i 1 dołączono do wyprowadzeń zewnętrznych. Trzeci układ czasowy służy do realizacji wewnętrznych zależności czasowych (może być używany jako dzielnik częstotliwości dla pozostałych dwóch układów czasowych lub jako źródło żądania DMA). Każdy z układów może być źródłem przerwania. Wyprowadzenia TMR IN0 i TMR IN1 umożliwiają wprowadzenie do układów czasowych o numerach 0 i 1 impulsów zewnętrznych (zliczane są zmiany z 0 na 1) lub sygnałów sterujących przy zliczaniu wewnętrznych impulsów zegarowych. Wyprowadzenia TMR OUT0 i TMR OUT1 umożliwiają wyprowadzenie sygnałów wyjściowych z układów czasowych o numerach odpowiednio 0 i 1.

Mikroprocesory 80186/188 zawierają również *wewnętrzny dwukanałowy sterownik* DMA – rys. 2.28. Zewnętrzne sygnały żądania bezpośredniego dostępu do

magistrali DRQ0 i DRQ1 doprowadza się do tych kanałów przez wyprowadzenia o tych samych nazwach. Każdy z kanałów zawiera 20-bitowe wskaźniki adresu źródła i przeznaczenia (złożone z dwóch rejestrów 16-bitowych; w rejestrze bardziej znaczącym wykorzystuje się tylko 4 mniej znaczące bity). Wskaźniki te mogą być zmniejszane lub zwiększane o jeden po każdym przesłaniu bajtu lub o dwa po przesłaniu słowa bądź też nie zmieniane. Wskazują one zarówno przestrzeń adresową pamięci, jak i układów we-wy. W wewnętrznym kontrolerze DMA nie dysponuje się jawnymi sygnałami potwierdzania zwalniania magistrali przez mikroprocesor. Ich funkcję mogą pełnić sygnały odczytu z wymaganego źródła lub zapisu do wymaganego przeznaczenia bądź sygnały wyboru danej przestrzeni pamięci lub układu we-wy, jeżeli wskaźniki adresowe źródła i przeznaczenia będą zaprogramowane na te właśnie przestrzenie. Każdy z kanałów może wygenerować lub nie (zależnie od zaprogramowania) przerwanie po zakończeniu przesłania (wyzerowaniu licznika długości przesyłanego bloku).



Rys. 2.28. Wewnętrzny sterownik DMA

Jak już wspomniano wcześniej, mikroprocesory 80186 80188 wyposażono w wewnętrzny układ wyboru bloków pamięci i układów we-wy. Do wyboru bloków pamięci można wykorzystać 6 sygnałów UCS, LCS, MCS3...MCS0.

Sygnał $\overline{\text{UCS}}$ (ang. *upper CS*) umożliwia (poziomem niskim) wybranie górnej przestrzeni adresowej pamięci, wykorzystywanej zwykle przez oprogramowanie systemowe, obejmującej obszar:

FFC00H ... FFFFFH – 1 KB;
FF800H ... FFFFFH – 2 KB;
FF000H ... FFFFFH – 4 KB;
FE000H ... FFFFFH – 8 KB
FC000H ... FFFFFH – 16 KB;
F8000H ... FFFFFH – 32 KB;
F0000H ... FFFFFH – 64 KB;
E0000H ... FFFFFH – 128 KB;
C0000H ... FFFFFH – 256 KB,

zależnie od zaprogramowania (zawartości rejestru UMCS – ang. *upper memory CS register*).

Sygnał $\overline{\text{LCS}}$ (ang. *lower CS*) zapewnia (poziomem niskim) wybranie dolnej przestrzeni adresowej pamięci obejmującej obszar od 00000H do: 0003FFH (1 KB); 007FFH (2 KB); 00FFFH (4 KB); 01FFFH (8 KB); 03FFFH (16 KB); 07FFFH (32 KB); 0FFFFH (64 KB); 1FFFFH (128 KB) lub 3FFFFH (256 KB), zależnie od zaprogramowania (zawartości rejestru LMCS – ang. *lower memory CS register*).

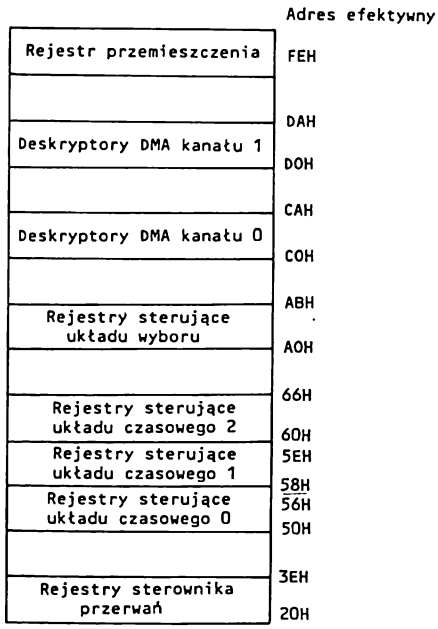
Sygnały $\overline{\text{MCS3}}$ ... $\overline{\text{MCS0}}$ (ang. *mid-range CS*) umożliwiają (poziomem niskim) wybranie czterech przestrzeni adresowych dla użytkownika o rozmiarach 2^n KB przy czym $n = 1, 2, ..., 7$ czyli 2, 4, 8, 16, 32, 64 lub 128 KB, zależnie od zaprogramowania (zawartości rejestru MPCS).

Obszary dla użytkownika mogą być umieszczone w dowolnym miejscu 1-megabajtowej przestrzeni adresowej pamięci z wyjątkiem obszarów, wskazywanych przez sygnały $\overline{\text{UCS}}$ i $\overline{\text{LCS}}$. Adresy bazowe tych obszarów (bloków) pamięci ustala się w programie (zawartością rejestru MMCS). Wszystkie 4 obszary użytkownika tworzą zawsze ciągłą przestrzeń adresową, poczynając od adresu będącego wielokrotnością 4×2^n (np. dla $2^n = 8$ KB będą to adresy 0H, 8000H, 10000H, 18000H itd.). Sygnały wyboru układów we-wy $\overline{\text{PCS4}}$ ... $\overline{\text{PCS0}}$, $\overline{\text{PCS5/A1}}$ i $\overline{\text{PCS6/A2}}$ umożliwiają wybranie 128-bajtowych obszarów w przestrzeni adresowej we-wy (64 KB) lub przestrzeni adresowej pamięci (1 MB), poczynając od zaprogramowanych adresów początkowych (bazowych) z odstępem co 1 KB (10 mniej znaczących bitów adresu bazowego ma stale wartość 0). Adres bazowy określa się zawartością rejestru PACS. Jeżeli przyjąć, że sygnałem $\overline{\text{PCS0}}$ wybiera się obszar 128-bajtowy od zadanego adresu bazowego (AB), to sygnałem $\overline{\text{PCS1}}$ wybiera się obszar od AB + 128 do AB + 255 itd., aż do sygnału $\overline{\text{PCS6}}$, którym wybiera się obszar od AB + 768 do AB + 895.

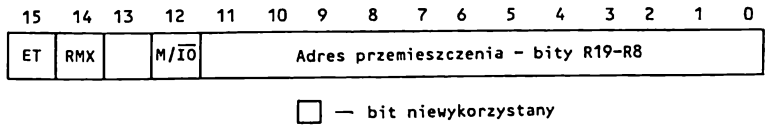
Wyprowadzenia $\overline{\text{PCS6}}$/A2 i $\overline{\text{PCS5}}$/A1 mają podwójne znaczenie – mogą być zarówno liniami wyboru układów we-wy, jak i liniami wyboru wewnętrznych rejestrów w 8-bitowych układach we-wy. Na wyprowadzeniach tych wówczas pojawiają się bity adresowe A2, A1, które można bezpośrednio użyć jako adresy A1, A0 układów we-wy. Funkcje wyprowadzeń $\overline{\text{ACS6/A2}}$ i $\overline{\text{PCS6/A1}}$ deklaruje się w programie, ustawiając odpowiednią wartość bitu EX (b7) w rejestrze MPCS. Gdy bit EX = 0, na wyprowadzeniach tych pojawiają się bity adresowe A2, A1, a gdy bit EX = 1, na wyprowadzeniach tych pojawiają się sygnały wyboru układów we-wy $\overline{\text{PCS6}}$ i $\overline{\text{PCS5}}$.

Mikroprocesory 80186/188 mogą generować wewnętrzny sygnał READY dla każdej linii wyboru pamięci lub układu we–wy oraz wtrącić od 0 do 3 taktów oczekiwania T_w. Liczbę wtrącanych taktów T_w deklaruje się na bitach R1, R0 (b1, b0) rejestru PACS dla sygnałów $\overline{PCS3}$...$\overline{PCS0}$, rejestru MPCS dla sygnałów $\overline{PCS6}$...$\overline{PCS4}$, rejestru UMCS dla sygnału $\overline{UCS}$, rejestru LMCS dla sygnału $\overline{LCS}$ i rejestru MMCS dla sygnałów $\overline{MCS3}$...$\overline{0}$. Bit R2 (b2) tych rejestrów wskazuje, czy zewnętrzne sygnały READY (SRDY, ARDY) są ignorowane (R2 = 1), czy też nie (R2 = 0). Przy czym, przy odwołaniach do wewnętrznych układów we–wy, zewnętrzne sygnały READY są zawsze ignorowane.

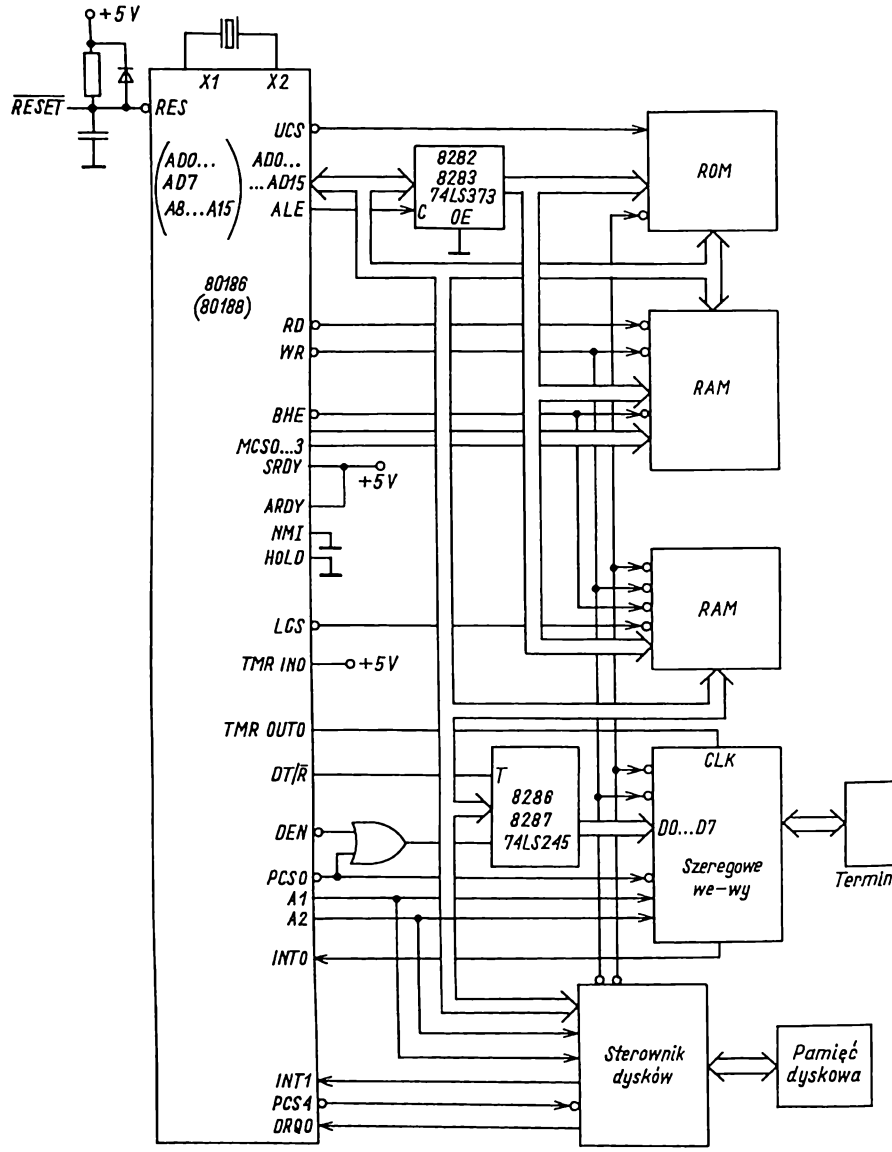
Wszystkie wewnętrzne układy we–wy w mikroprocesorach 80186/188 programuje się i kontroluje za pomocą 16–bitowych rejestrów (sterujących), umieszczonych w 256–bajtowym obszarze, który może znajdować się w przestrzeni adresowej pamięci lub układu we–wy. Adres efektywny (offset) rejestrów w tym obszarze jest stały – rys. 2.29. Adres początkowy (bazowy) tego obszaru deklaruje się zawartością tzw. *rejestru przemieszczania* – rys. 2.30.



Rys. 2.29. Wykaz adresów rejestrów sterujących w 256–bajtowym obszarze



Rys. 2.30. Zawartość rejestru przemieszczenia



Rys. 2.31. Przykład mikrokomputera z mikroprocesorem 80186 (80188)

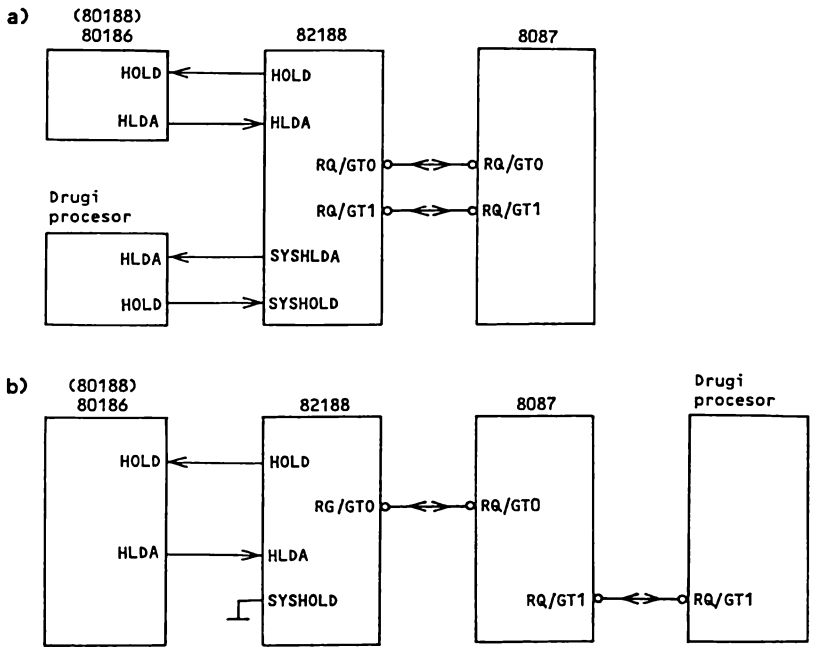
Podczas zerowania rejestr ten zapisuje się taką zawartością, aby adres początkowy obszaru rejestrów sterowania wynosił FF00H w przestrzeni adresowej we-wy. Adres początkowy można zmienić podczas inicjo-wania systemu, zmieniając

zawartość rejestru przemieszczania. Pozostałe bity rejestru przemieszczania mają następujące znaczenie:

- ET = 1 oznacza, że "ESC trap" występuje, a ET = 1, że nie występuje;
- RMX = 0 oznacza normalny tryb pracy sterownika przerwań, a RMX = 1 – tryb IRMX;
- M/$\overline{IO}$ = 1 oznacza, że blok rejestrów jest umieszczony w przestrzeni adresowej pamięci, a M/$\overline{IO}$ = 0, że w przestrzeni adresowej we–wy.

Strukturę przykładowego mikrokomputera z mikroprocesorem 80186/188 przedstawiono na rys. 2.31.

Rozszerzenie mikrokomputera z mikroprocesorem 80186 lub 80188 o koprocesory można łatwo zrealizować, stosując bardziej rozbudowany sterownik magistrali 82188 – rys. 2.32. W drugim procesorze, na rys. 2.32a, żąda się dostępu do magistrali, ustawiając sygnał SYSHOLD = 1 w sterowniku magistrali. Ze sterownika magistrali, za pośrednictwem sygnału HOLD, przekazuje się to żądanie do procesora głównego, w którym potwierdza się jej zwolnienie sygnałem HLDA = 1. Na tej podstawie w sterowniku magistrali ustawia sygnał SYSHLDA = 1 i przyznaje magistralę drugiemu procesorowi.



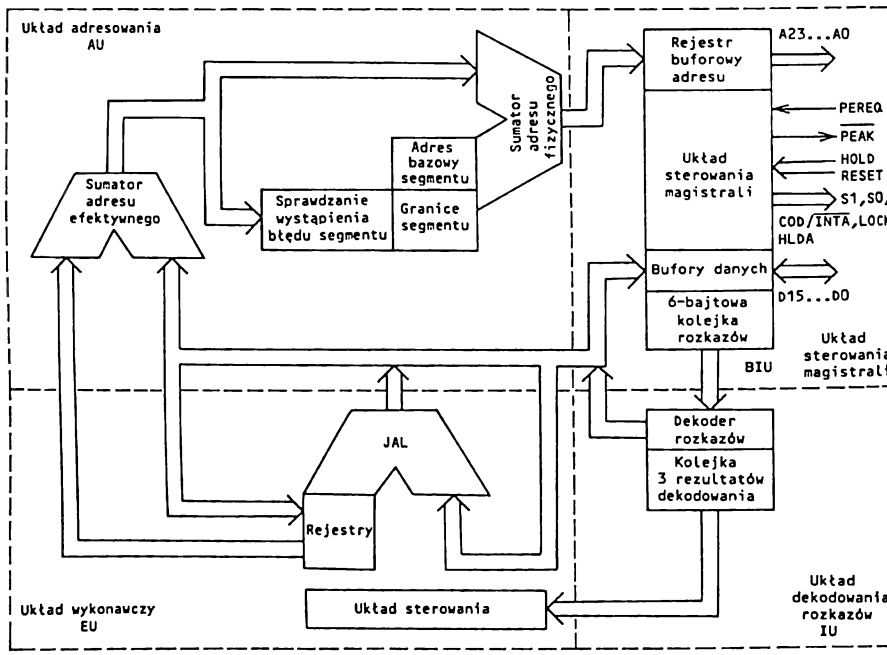
Rys. 2.32. Architektura jednostki centralnej z dwoma koprocesorami

Na rysunku 2.32b przedstawiono możliwość dołączenia dodatkowych koprocesorów (procesorów) za pośrednictwem dwukierunkowych linii $\overline{RQ}/\overline{GT}$. Dostęp do magistrali uzyskuje się na zasadzie podobnej do stosowanej przy współpracy procesorów i koprocesorów z mikroprocesorem 8086/88 (por. p. 2.1).

W konstrukcji mikrokomputera klasy IBM PC/XT (patrz p.3.1) przy zastosowaniu mikroprocesora 80188 jest wymagana znacznie mniejsza liczba układów scalonych niż przy zastosowaniu mikroprocesora 8088.

2.3. Mikroprocesor 80286

Mikroprocesor 80286 jest mikroprocesorem 16-bitowym nowej generacji o znacznie bogatszych właściwościach funkcjonalnych niż jego poprzedniki 8086, 8088 czy 80186, 80188.



Rys. 2.33. Struktura wewnętrzna mikroprocesora 80286

Dodatkowe właściwości to przede wszystkim

- możliwość pracy wielozadaniowej (wieloprogramowej);
- możliwość pracy w trybie adresowania rzeczywistego (ang. *real address mode*) lub adresowania wirtualnego z ochroną zasobów (ang. *protected virtual address mode*);

- wbudowany układ zarządzania pamięcią MMU (ang. *memory menagement unit*), zapewniający użytkownikowi 1–gigabajtową (1 GB) przestrzeń adresową pamięci wirtualnej dla każdego zadania i 16–megabajtową (16 MB) przestrzeń adresową pamięci fizycznej;
- większa wydajność obliczeniowa wyrażana w milionach rozkazów, wykonywanych w czasie 1 sekundy, czyli w tzw. *MIPS*–ach (ang. *million instruction per second*).

Przykładowo, w wersji mikroprocesora 80286 o częstotliwości zegara 10 MHz zapewniono co najmniej 5–krotnie większą wydajność obliczeniową niż w wersji procesora 8086 o częstotliwości zegara 5 MHz. Produkuje się ponadto mikroprocesory 80286, pracujące z zegarem 6; 8; 12,5; 16 i 20 MHz.

Z punktu widzenia użytkownika mikroprocesor 80286, pracujący w trybie adresowania rzeczywistego, jest w pełni zgodny z mikroprocesorami 8086/88 i 80186/188 w zakresie wynikowego kodu programu i zapewnia również dostęp do 1–megabajtowej przestrzeni adresowej pamięci fizycznej. W trybie adresowania wirtualnego z ochroną zasobów istnieje zgodność z mikroprocesorami 8086/88 i 80186/188 jedynie na poziomie programów źródłowych. Oznacza to, że muszą one być ponownie tłumaczone przez przygotowane dla mikroprocesora 80286 translatory.

Strukturę wewnętrzną mikroprocesora 80286 przedstawiono na rys. 2.33, wyróżnić w niej można:
– układ sterowania magistrali **BIU** (ang. *bus interface unit*);
– układ rozkazowy **IU** (ang. *instruction unit*);
– układ wykonawczy **EU** (ang. *execution unit*);
– układ adresowania **AU** (ang. *address unit*).

Układ sterowania magistrali umożliwia komunikację z pamięcią i układami we–wy za pośrednictwem sygnałów adresowych, danych i sterujących (tylko sygnały INTR, NMI, $\overline{\text{ERROR}}$ i $\overline{\text{BUSY}}$ spośród sygnałów magistrali mikroprocesora oddziałują wprost na układ sterowania, znajdujący się w układzie wykonawczym).

Układ rozkazowy zapewnia dekodowanie rozkazów pobieranych z 6–bajtowej kolejki rozkazów (znajdującej się w układzie magistrali) i umieszczenie wyników dekodowania w kolejce o pojemności, umożliwiającej przechowywanie wyników dekodowania 3 rozkazów. Uzyskane w ten sposób wewnętrzne sygnały sterujące oddziałują na pozostałe układy mikroprocesora.

Układ wykonawczy wykonuje operacje arytmetyczno–logiczne z użyciem jednostki JAL (ALU – ang. *arithmetic–logic unit*).

Układ adresowania wylicza składowe adresu i wyznacza adres fizyczny, który jest przekazywany na linie adresowe poprzez układ rejestrów buforowych.

Wszystkie te układy dzięki rejestrom buforowym i kolejkom mogą pracować równolegle. Można więc powiedzieć, że mikroprocesor 80286 umożliwia *przetwarzanie potokowe* (ang. *pipeline processing*), co znacznie przyspiesza realizację programów. Przedstawimy to na przykładzie wykonywania rozkazów przetwarzających dane zawarte w rejestrach roboczych i umieszczających wynik również w tych rejestrach.
W cyklu rozkazowym takiego rozkazu wyróżnia się 4 fazy
– obliczanie adresu kodu rozkazu (przez AU) – OA_i;
– pobranie kodu rozkazu (przez BIU) – PR_i;
– dekodowanie kodu rozkazu (przez IU) – DR_i;

– wykonanie rozkazu (przez EU) – WR_i
przy czym i = 1, 2, 3, ... – numer kolejnego rozkazu w programie.

Przy *przetwarzaniu sekwencyjnym* (rys. 2.34a) te 4 fazy są wykonywane szeregowo dla kolejnych rozkazów. Oznacza to, że przy przetwarzaniu sekwencyjnym, w danej chwili tylko jeden z układów BIU, IU, EU lub AU wykonywałby odpowiadającą mu fazę rozkazu, a pozostałe byłyby w stanie oczekiwania.

a)

AU	BIU	IU	EU
OA_i	PR_i	DR_i	WR_i

b)

AU	OA_i	OA_{i+1}	OA_{i+2}	OA_{i+3}
BIU	PR_{i-1}	PR_i	PR_{i+1}	PR_{i+2}
IU	DR_{i-2}	DR_{i-1}	DR_i	DR_{i+1}
EU	WR_{i-3}	WR_{i-2}	WR_{i-1}	WR_i

Rys. 2.34. Przetwarzanie danych: a) sekwencyjne; b) potokowe

Przy *przetwarzaniu potokowym* (rys. 2.34b), w danej chwili każdy z układów AU, BIU, IU, EU wykonuje odpowiadającą mu fazę rozkazu, przy czym są to fazy różnych rozkazów dla każdego z tych układów. Układy te pracują więc równolegle, dzięki czemu rozpatrywane rozkazy są wykonywane 4-krotnie szybciej niż przy przetwarzaniu sekwencyjnym. Przy wykonywaniu rozkazów, wymagających dodatkowych odwołań do pamięci, szybkość ta nieco maleje. Na przykład przy rozkazach, przesyłających zawartości rejestru do komórki pamięci, zysk czasowy nie będzie się wyrażał stosunkiem 4 : 1, lecz 6 : 2 (rys. 2.35).

a)

AU	BIU	IU	EU	AU	BIU
OA_i	PR_i	DR_i	WR_i	OA'_i	ZD_i

Zapis danej

b)

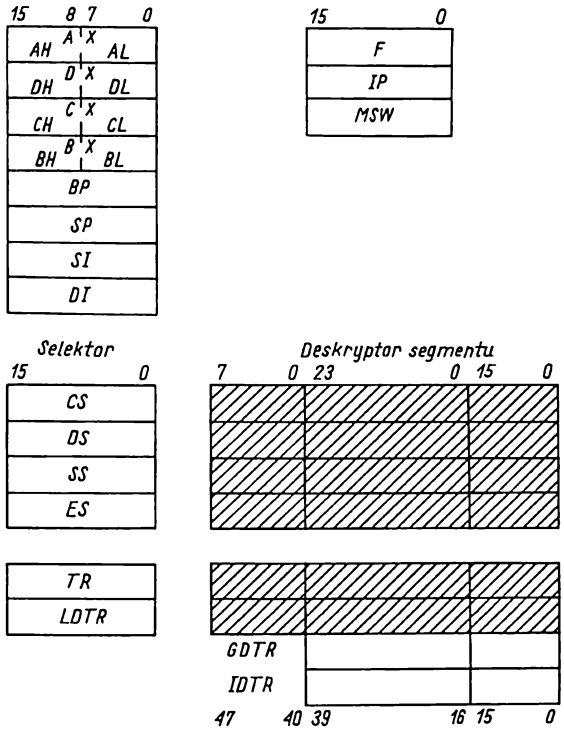
AU	OA_i	OA'_{i-1}	OA_{i+1}	OA'_i	OA_{i+2}	OA'_{i+2}
BIU	ZD_{i-2}	PR_i	ZD_{i-1}	PR_{i+1}	ZD_i	PR_{i+2}
IU	DR_{i-1}	—	DR_i	—	DR_{i+1}	—
EU	—	WR_{i-1}	—	WR_i	—	WR_{i+1}

Rys. 2.35. Przetwarzanie: a) sekwencyjne; b) potokowe

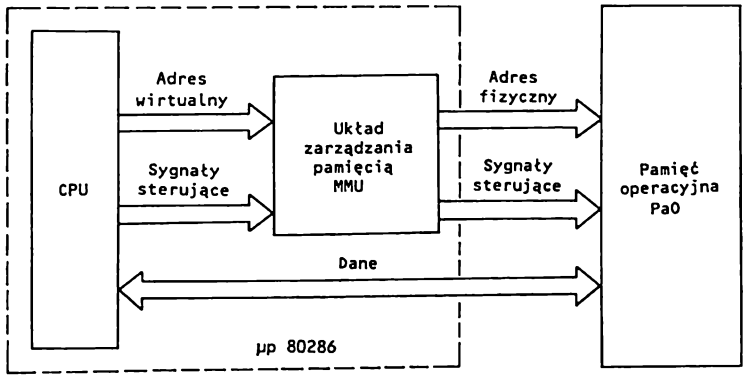
Jak już wspomniano, mikroprocesor 80286 może pracować w trybie adresowania rzeczywistego bądź wirtualnego z ochroną zasobów. Podczas pracy w trybie adresowania rzeczywistego mikroprocesor ten zapewnia użytkownikowi takie same możliwości funkcjonalne, jak mikroprocesor 8086 (patrz p. 2.1).

Dostępne dla programisty są takie same rejestry robocze i specjalne, jak w mikroprocesorze 8086 (porównaj rys. 2.36 i 2.1). Podczas pracy w trybie adresowania wirtualnego procesor 80286 zapewnia 1-gigabajtową przestrzeń adresowania wirtualnego i znacznie mniejszą, bo 16-magabajtową przestrzeń adresowania fizycznego. Przy opracowaniu programu użytkownik, posługując się adresami wirtualnymi (logicznymi)

dysponuje pełną 1–gigabajtową przestrzenią adresową. Podczas realizowania programu, specjalny układ zarządzania pamięcią MMU przekształca adres wirtualny w adres fizyczny przy każdym odwołaniu do pamięci operacyjnej – rys. 2.37.



Rys. 2.36. Rejestry mikroprocesora 80286



Rys. 2.37. Sposób przekształcenia adresu wirtualnego w adres fizyczny

W mikroprocesorze 80286, w przeciwieństwie do innych mikroprocesorów 16-bitowych tej klasy, MMU stanowi część układu adresowania mikroprocesora, umieszczoną z nim w tym samym układzie scalonym. Upraszcza to konstrukcję jednostki centralnej i przyspiesza sterowanie operacyjnej pamięci fizycznej. Pamięć operacyjna w rozpatrywanym przypadku ma również organizację bajtową i jest segmentowana. Inaczej jest tu jednak interpretowana zawartość rejestrów segmentów (CS, DS, SS, ES); pojawiają się również dodatkowe rejestry dostępne dla programisty oraz rejestry niedostępne (tworzące szybką pamięć typu "Cache"). Rejestry niedostępne dla programisty zakreskowano na rys. 2.36. Odgrywają one istotną rolę zarówno przy translacji adresu wirtualnego na fizyczny, jak i przy przełączaniu zadań przy pracy wielozadaniowej. Ponieważ z założenia przestrzeń adresowa pamięci wirualnej jest większa niż przestrzeń adresowa pamięci fizycznej, może się zdarzyć, że przy wykonywaniu programu nastąpi odwołanie do segmentu, którego aktualnie nie ma w operacyjnej pamięci fizycznej. System operacyjny przy współpracy z układem MMU wykryje taką sytuację i zainicjuje wczytanie z zewnętrznej pamięci dyskowej żądanego przez program segmentu do pamięci fizycznej (do obszaru wolnego bądź obszaru zajętego przez inny aktualnie niewykorzystywany segment). Jeżeli w pamięci fizycznej brak wolnego obszaru, to przy wczytywaniu żądanego segmentu mogą wystąpić dwa przypadki:

- przeznaczony do usunięcia segment nie był modyfikowany, nie trzeba go więc chować w zewnętrznej pamięci dyskowej przed wczytaniem na jego miejsce brakującego segmentu;
- przeznaczony do usunięcia segment był modyfikowany, należy go więc przesłać do zewnętrznej pamięci dyskowej przed wczytaniem na jego miejsce brakującego segmentu.

Dzięki takiej organizacji pamięci wirtualnej użytkownik ma wrażenie, że współpracuje z większą pamięcią niż ta, która fizycznie występuje.

Adres wirtualny składa się z numeru segmentu i *adresu efektywnego* (zależnego od rozkazu) określającego położenie bajtu pamięci w segmencie. Jeżeli założymy, że numer segmentu jest określony na m-bitach, a adres efektywny (offset) na n-bitach, to 2^m określa maksymalną liczbę segmentów w programie, a 2^n – maksymalną pojemność segmentu w bajtach. Daje to użytkownikowi pamięć wirtualną o pojemności 2^{m+n} bajtów.

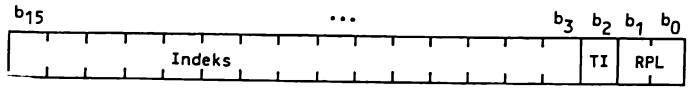
W mikroprocesorze 80286 $m = 14$, a $n = 16$, co daje $2^{30} = 1$ GB przestrzeni adresowania pamięci wirtualnej.

Numer segmentu jest zapisany w jednym z rejestrów segmentów (spełniających w trybie adresowania wirtualnego rolę selektora segmentu). Zawartość rejestru segmentu (selektora segmentu) – rys. 2.38 – jest w tym przypadku interpretowana następująco:

bity b_1, b_0 – RPL (ang. *requested privilege leved*) – wskazują żądany poziom uprzywilejowania selektora (00 – poziom najwyższy);

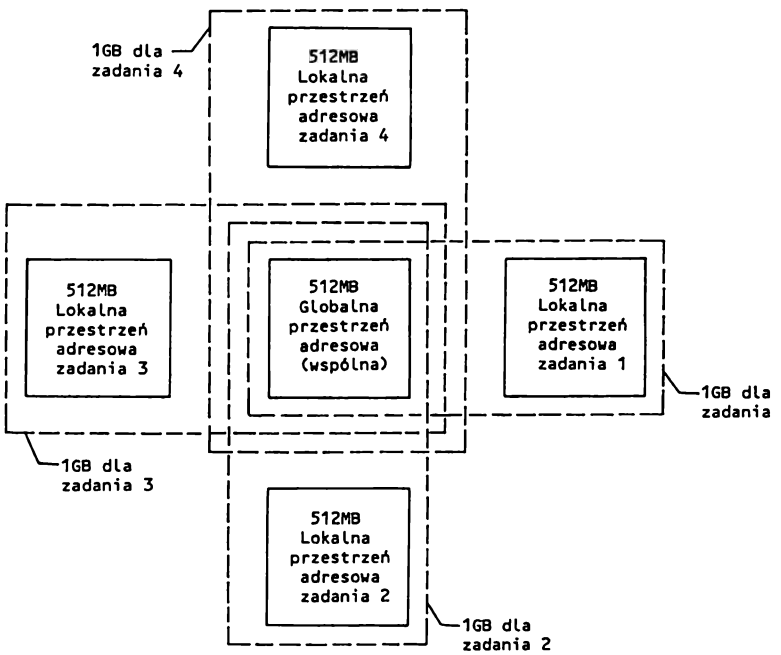
bit b_2 – TI (ang. *table indicator*) – wskazuje tablicę deskryptorów lokalnych, gdy TI = 1 bądź globalnych, gdy TI = 0 w pamięci wirtualnej;

bity b_{15} ... b_3 – stanowią 13-bitowy indeks (numer segmentu) w lokalnej bądź globalnej 512-MB przestrzeni adresowej pamięci wirtualnej.

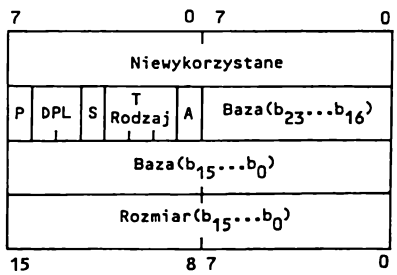


Rys. 2.38. Zawartość rejestru segmentu

Przestrzeń adresową pamięci wirtualnej podzielono na lokalną (512 MB), oddzielną dla każdego zadania oraz globalną (512 MB), wspólną dla wszystkich zadań (rys. 2.39).



Rys. 2.39. Globalna i lokalna przestrzeń adresowa



Rys. 2.40. Format deskryptora segmentu danych lub programu

W układzie MMU, umieszczonym w układzie adresowania pamięci, na podstawie adresu wirtualnego wyznacza się 24-bitowy adres fizyczny umożliwiający adresowanie 16-megabajtowej przestrzeni adresowej.

W mikroprocesorze korzysta się przy tym z dodatkowych rejestrów TR, LDTR, GDTR, IDTR i MSW, dostępnych dla programisty oraz rejestrów niedostępnych, tzw. *rejestrów deskryptorów segmentów*. Zawartość rejestrów niedostępnych jest zmieniana automatycznie przez MMU. Rejestry deskryptorów segmentów danych, programu i stosu (patrz rys. 2.36) stanowią rozszerzenie rejestrów segmentów (CS, DS, SS i ES) i służą do przechowywania 24-bitowego *adresu bazowego* (ang. *segment base address*), wskazującego adres początkowy segmentu (bazę), 16-bitowego rozmiaru segmentu (ang. *limit*) i 8-bitowego pola atrybutów segmentu, zwanego także bajtem reguł dostępu (ang. *access rights byte*). Pełny deskryptor segmentu danych lub programu – rys. 2.40 – zawiera 8 bajtów, z tym że w mikroprocesorze 80286 dwa z nich nie są wykorzystywane (firma Intel zarezerwowała je dla potrzeb mikroprocesorów nowych generacji, w tym dla 32-bitowego mikroprocesora 80386).

Nie wykorzystywane bity deskryptora w mikroprocesorze 80286 ustawiono na zero. Adres bazowy segmentu wskazuje początek segmentu, a adres efektywny – położenie argumentu w segmencie względem jego początku. W przypadku segmentu programu położenie (offset) rozkazu określa się zawartością licznika rozkazów IP, a w przypadku segmentu stosu położenie (offset) danej określa się np. zawartością wskaźnika stosu SP lub rejestru BP.

Bajt atrybutów zawiera pola: **A** (ang. *accessed*), **T** (ang. *type*), **S** (ang. *segment descriptor*), **DPL** (ang. *descriptor privilege level*) *i* **P** (ang. *present*).

Pole **A** wskazuje, że nie było odwołania do segmentu danych (w tym stosu) lub programu znajdującego się w pamięci (A = 0) bądź, że selektor segmentu załadowano do rejestru segmentu (było odwołanie do segmentu) lub użyto go w rozkazach testujących selektor (A = 1). Bit A wykorzystuje się w systemie operacyjnym do określenia, który z segmentów znajdujących się w pamięci operacyjnej należy usunąć (w przypadku braku wolnej przestrzeni w pamięci i konieczności załadowania żądanego w programie segmentu). Odbywa się to w ten sposób, że w oprogramowaniu systemowym, co określony przedział czasu, testuje się wartość tego bitu i przy A = 1 zwiększa się zawartość tablicy, informującej o liczbie odwołań do danego segmentu oraz zeruje ten bit (A = 0). Na podstawie zapisu w tablicy układ zarządzania pamięcią w razie potrzeby usuwa z pamięci operacyjnej segment, do którego było najmniej odwołań.

Pole **T** (3-bitowe) określa reguły dostępu do segmentu i jest inaczej interpretowane w deskryptorze segmentu danych niż w deskryptorze segmentu programu. Rozróżnienie deskryptorów tych segmentów odbywa się za pomocą bitu b_3 oznaczonego symbolem E (ang. *executable*).

Dla *deskryptora segmentu danych* bądź *stosu* (E = 0 i S = 1)

bit b_2 = ED (ang. *expansion direction*) wskazuje kierunek rozszerzania segmentu, przy czym:
ED = 0 – oznacza rozszerzenie segmentu w kierunku rosnących adre-sów. Wartość offsetu w tym przypadku musi być mniejsza bądź równa wartości pola rozmiaru deskryptora. Przypadek ten

odnosi się do segmentu danych;
ED = 1 – oznacza rozszerzenie deskryptora w kierunku malejących adresów. Wartość offsetu w tym przypadku musi być większa niż wartość pola rozmiaru deskryptora. Przypadek ten odnosi się do segmentu stosu;

bit b_1 = W (ang. *writeable*) wskazuje, czy segment danych bądź stosu można zapisywać (modyfikować) – W = 1, czy nie można – W = 0.

Dla *deskryptora segmentu programu* (E = 1, S = 1)

bit b_2 = C = 1 (ang. *conforming*) oznacza, że program zawarty w segmencie programu można wykonywać tylko wówczas, gdy bieżący poziom uprzywilejowania programu zadania CPL (ang. *current privilege level*) jest niższy bądź równy poziomowi uprzywilejowania deskryptora DPL i poziom CPL pozostaje niezmieniony. Oznacza to, że liczbowo CPL ≥ DPL;

bit b_1 = R (ang. *readable*) wskazuje, czy można czytać segment programu (R = 1), czy też nie (R = 0).

Pole **S** wskazuje, czy jest to deskryptor segmentu danych lub programu – S = 1, czy też deskryptor segmentu systemowego (ang. *system segment descriptor*) lub bramki (ang. *gate descriptor*) – S = 0.

Pole **DPL** określa, na bitach b_6 i b_5, poziom uprzywilejowania deskryptora, testowany przez MMU. Poziom ten określa, czy w zadaniu można korzystać z deskryptora segmentu (00 – oznacza najwyższy, a 11 – najniższy poziom uprzywilejowania deskryptora).

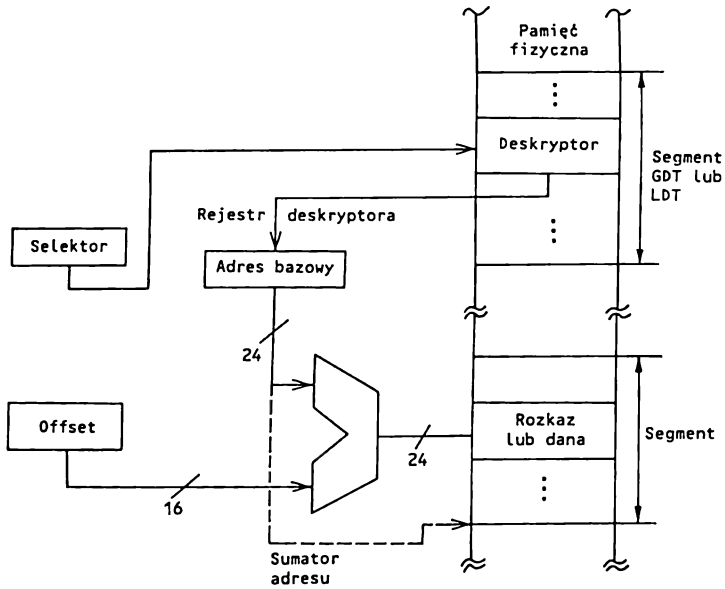
Wartość bitu pola **P** = 1 wskazuje, że zawartość deskryptora jest istotna i odpowiadający mu segment znajduje się w pamięci operacyjnej, a wartość bitu P = 0 – sytuację odwrotną.

Deskryptory segmentów, występujących w globalnej przestrzeni adresowej, tworzą tzw. *tablicę deskryptorów globalnych* **GDT** (ang. *global descriptor table*). Tablica ta może zawierać do 8192 deskryptorów (co odpowiada maksymalnej pojemności segmentu) dostępnych dla wszystkich zadań. Mogą znajdować się w niej wszystkie rodzaje deskryptorów oprócz *deskryptorów przerwań* i *pułapki* (ang. *trap*). Adres bazowy (początkowy) segmentu, tworzącego tablicę GDT oraz jego rozmiar, deklaruje się rozkazem *LGDT* (patrz p. 4.1) w rejestrze tablicy deskryptorów globalnych GDTR (ang. *global descriptor table register*) – patrz rys. 2.36.

Oprócz tablicy deskryptorów globalnych występują *tablice deskryptorów lokalnych* **LDT** (ang. *local descriptor table*) w ilości odpowiadającej ilości wykonywanych zadań (jedna tablica deskryptorów lokalnych na jedno zadanie). Tablica LDT tworzy segment, który może zawierać, podobnie jak w tablicy GDT, do 8192 deskryptorów, dostępnych dla danego zadania. Do określenia jej rozmiaru i adresu bazowego jest niezbędny odpowiadający jej deskryptor. Deskryptory tablic deskryptorów lokalnych znajdują się w tablicy deskryptorów globalnych. Ich położenie w tablicy deskryptorów globalnych określa rejestr LDTR (ang. *local descriptor table register*) spełniający funkcje selektora segmentu zawierającego LDT.

Deskryptor LDT, załadowany do rejestru deskryptora segmentu LDT (stanowiącego rozszerzenie LDTR), zawiera adres bazowy tablicy deskryptorów lokalnych, a położenie (offset) deskryptora segmentu programu, danych bądź stosu w tablicy deskryptorów lokalnych określa odpowiedni selektor (CS, DS, ES lub SS).

Wyznaczony w ten sposób deskryptor segmentu (CS, DS, ES lub SS) jest automatycznie ładowany przez układ MMU do rejestru deskryptora (CS, DS, ES lub SS). Zawarty w nim 24-bitowy adres bazowy wskazuje początek odpowiedniego segmentu programu, danych lub stosu. Położenie (offset) rozkazu lub danej w segmencie określa zawartość wskaźnika rozkazów IP, wskaźnika stosu SP lub adres efektywny, zależny od trybu adresowania rozkazu. Proces wyznaczania adresu fizycznego rozkazu lub danej w segmencie CS, DS, ES lub SS na podstawie adresu wirtualnego przedstawiono na rys. 2.41 i 2.42. Występujące na tym rysunku odwołania do pamięci fizycznej dotyczą komórek o adresach fizycznych, wyznaczonych w sumatorze (ang. *physical address adder*) na podstawie 24-bitowego adresu bazowego i 16-bitowego położenia w segmencie.

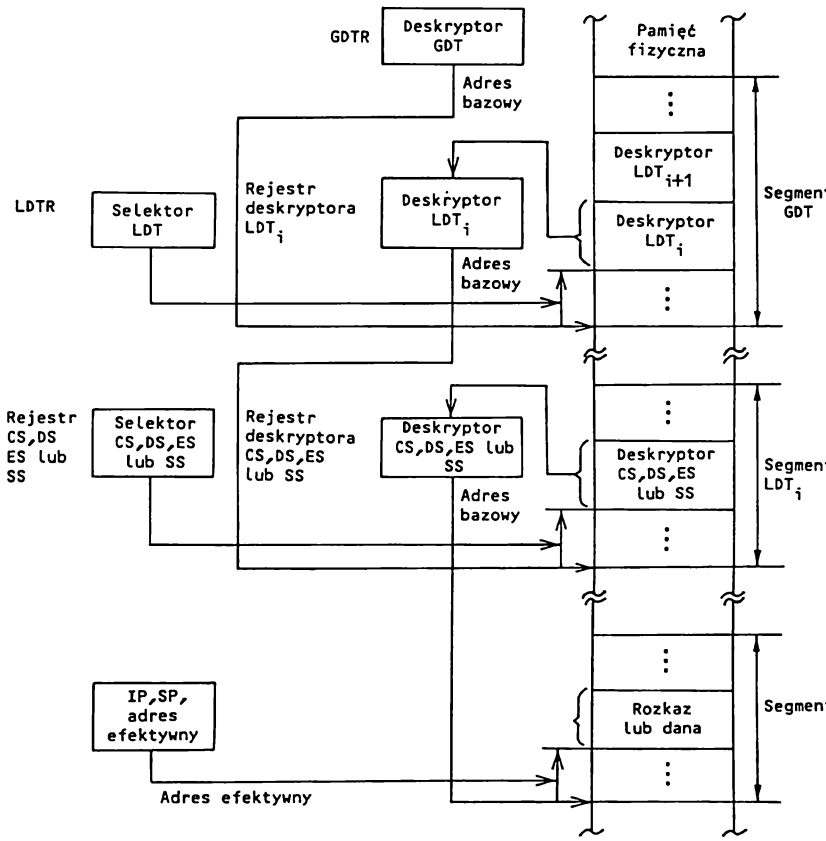


Rys. 2.41. Ogólna zasada określania adresu fizycznego na podstawie adresu wirtualnego

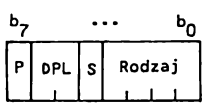
W mikroprocesorze 80286 oprócz deskryptorów segmentów programu, danych i stosu są również deskryptory segmentów systemowych, zawierających tablice deskryptorów lokalnych LDT lub określających stan wykonywanego zadania (ang. *task state segment descriptor*). Są to, tzw. *sterujące deskryptory systemowe* umieszczone w tablicy deskryptorów globalnych GDT. Ich format jest taki sam, jak deskryptorów segmentów danych i programu (patrz rys. 2.40), z tym, że poszczególne bity bajtu reguł dostępu mają nieco inne przeznaczenie (rys. 2.43).

Bajt reguł dostępu określa rodzaj deskryptora, jego stan i poziom uprzywilejowania. W deskryptorach tablic deskryptorów lokalnych nie używa się pola DPL. Cztery najmniej znaczące bity tego bajtu określają rodzaj deskryptora.

Stan bitów $b_3b_2b_1b_0$ = 0001 – wskazuje, że segment stanu zadania (TSS) jest dostępny;
0011 – wskazuje, że segment stanu zadania jest zajęty;
0010 – wskazuje, że jest to deskryptor tablicy deskryptorów lokalnych;



Rys. 2.42. Kolejne etapy wyznaczania adresu fizycznego na podstawie adresu wirtualnego w lokalnej przestrzeni adresowej



Rys. 2.43. Format bajtu reguł dostępu deskryptora systemowego

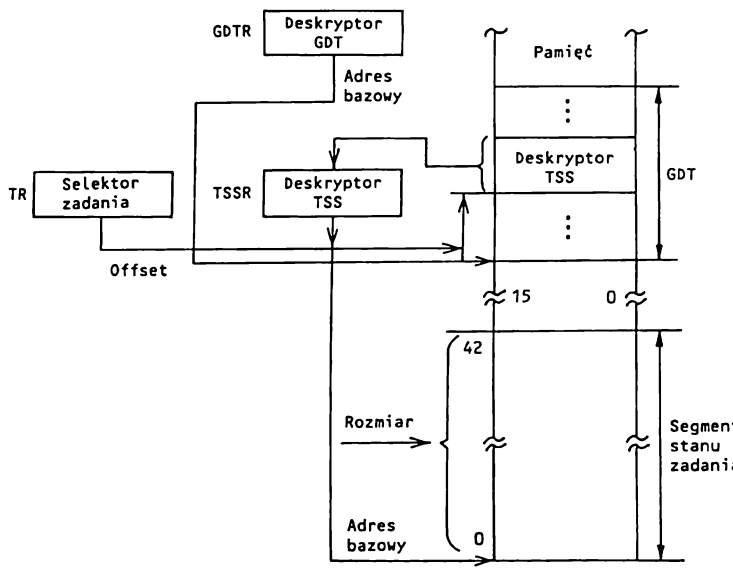
Stan bitu b_4 = S = 0 – wskazuje, że jest to deskryptor segmentu systemowego.

Bity b_6b_5 = DPL – (używane tylko w deskryptorze segmentu stanu zadania) wskazują poziom uprzywilejowania deskryptora, przy którym można go użyć.

Bit b_7 = P – ma takie samo znaczenie, jak jego odpowiednik w deskryptorach programu lub danych.

Segment stanu zadania wykorzystuje się do sterowania wielozadaniową pracą procesora 80286. Każde wykonywane zadanie ma swój segment stanu zadania. Ich deskryptory, podobnie jak deskryptory LDT, przechowuje się w tablicy deskryptorów globalnych (GDT). W segmencie stanu danego zadania TSS są przechowywane:

- informacje o powiązaniu z zadaniem wykonywanym bezpośrednio przed określonym zadaniem. Umożliwiają one między innymi powrót do poprzedniego zadania po zakończeniu obsługi przerwania. (Procedury obsługi przerwań, dzięki bardzo szybkiemu sprzętowemu przełączaniu zadań, można realizować jako odrębne zadania);
- zawartości początkowe wszystkich rejestrów, charakteryzujących stan procesora oraz wskaźników stosu dla 0–, 1– i 2–go poziomu uprzywilejowania;
- selektor tablicy deskryptorów lokalnych.



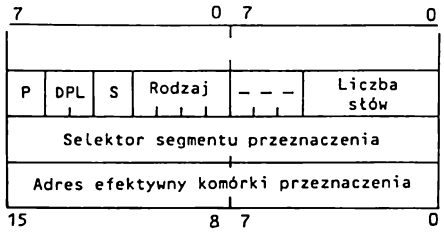
Rys. 2.44. Sposób wyznaczenia adresu bazowego segmentu stanu zadania

Sposób wyznaczania adresu bazowego segmentu stanu zadania przedstawiono na rys. 2.44. Przy przełączaniu się na inne zadanie mikroprocesor umieszcza stan wykonywanego zadania w segmencie TSS. Rejestr zadania (ang. *task register*) ładuje się w programie selektorem nowego zadania, który umożliwia zaadresowanie deskryptora

stanu nowego zadania. Deskryptor ten ładuje się do rejestru deskryptora stanu zadania, a stan początkowy nowego zadania – do odpowiednich rejestrów mikroprocesora, jednocześnie rejestr tablicy deskryptorów lokalnych (LDTR) jest ładowany selektorem lokalnej tablicy deskryptorów nowego zadania. Problemy związane z przełączaniem zadań będą przedstawione dokładniej w punkcie 4.2.

Oprócz omówionych deskryptorów w mikroprocesorze 80286 stosuje się deskryptory bramek (ang. *gate descriptors*). Istnieją 4 rodzaje deskryptorów bramek często zwane *bramkami*:
– wywołania (ang. *call*);
– zadania (ang. *task*);
– przerwania (ang. *interrupt*);
– pułapki (ang. *trap*).

Bramki wywołania stosuje się do wywoływania podprogramów i do zmiany poziomów uprzywilejowania, *bramki zadania* – do przełączania zadań, a *bramki przerwania* i *pułapki* – do określania adresów podprogramów obsługi przerwania. Przy użyciu bramki przerwania następuje zerowanie znacznika przerwania IF w rejestrze znaczników F, czyli zablokowanie przerwania. Bramka pułapki nie blokuje przerwań. Format bramki (deskryptora bramki) przedstawiono na rys. 2.45.



Rys. 2.45. Format deskryptora bramki

Położenie (offset) wraz z częścią indeksową selektora wskazuje adres przeznaczenia. Pięciobitowe pole liczby słów jest wykorzystywane tylko w deskryptorze bramki wywołania. Wskazuje ono liczbę słów (w naturalnym kodzie dwójkowym) automatycznie przesyłanych (przepisywanych) z poprzedniego do nowego stosu, odpowiadającego wywołanej procedurze przy zmianie poziomu uprzywilejowania zadania po rozkazie wywołania. Wynika to stąd, że w każdym zadaniu mogą występować 4 poziomy uprzywilejowania i każdemu z nich przyporządkowano odrębny stos. Pozostałe trzy bity tego pola nie są wykorzystywane.

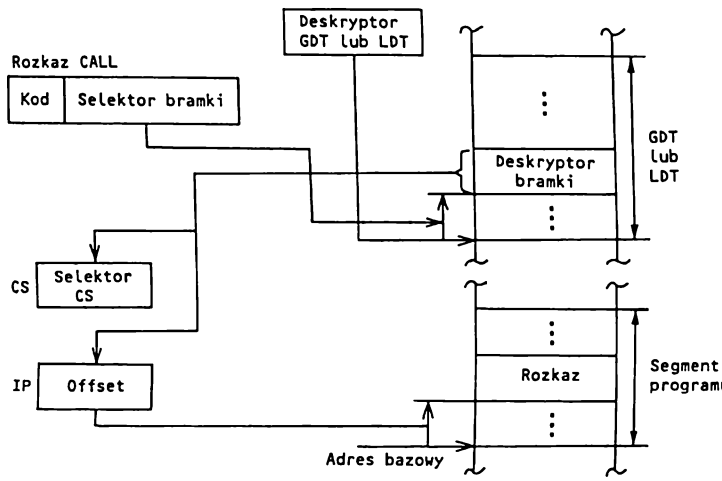
Pole bajtu reguł dostępu ma format podobny do formatu tego pola w deskryptorze systemowym. Cztery najmniej znaczące bity tego bajtu określają rodzaj bramki:

$b_3b_2b_1b_0$ = 0100 – wywołania;
0101 – zadania;
0110 – przerwania;
0111 – pułapki.

Pole S ma w deskryptorach bramek stale wartość 0. Pole DPL określa jeden z czterech (0, 1, 2, 3) poziomów uprzywilejowania deskryptora bramki. Pole P = 0

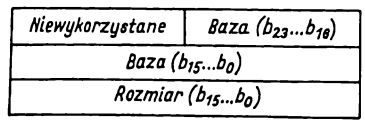
sygnalizuje, że zawartość deskryptora jest nieważna, a pole P = 1 wskazuje, że zawartość jest ważna i można go użyć do wyznaczania adresu komórki w pamięci fizycznej. Pozostałe dwa bajty, tak jak w przedstawionych wcześniej deskryptorach, nie są wykorzystywane.

Sposób wyznaczania początkowego adresu podprogramu, wywołanego rozkazem *CALL* za pomocą bramki wywołania, przedstawiono na rys. 2.46. Część indeksowa selektora (ładowana do CS) i położenie (offset) (ładowany do wskaźnika IP) umożliwiają wyznaczenie adresu początku podprogramu. Adres bazowy segmentu programu określa deskryptor wskazany przez część selektorową CS.



Rys. 2.46. Sposób wyznaczenia adresu początkowego podprogramu

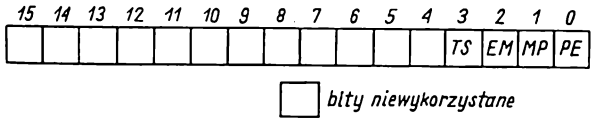
Rejestr tablicy deskryptorów przerwań IDTR (ang. *interrupt descriptor table register*) wskazuje segment w globalnej przestrzeni adresowej. W segmencie tym, tworzącym tablicę deskryptorów przerwań (ang. *interrupt descriptor table*), mogą występować deskryptory bramek zadania, przerwań i pułapki. Tablica IDT jest wykorzystywana przez układ MMU do określania do 256 początkowych adresów podprogramów obsługi przerwań. Format deskryptora zawartego w rejestrze IDTR przedstawiono na rys. 2.47 (jest on identyczny z formatem deskryptora, przechowywanego w GDTR, wskazującego tablicę deskryptorów globalnych). Występuje w nim tylko rozmiar segmentu i fizyczny adres bazowy.



Rys. 2.47. Format deskryptora ładowanego do rejestrów IDTR i GDTR

Bajtu reguł dostępu nie wykorzystuje się i powinien on mieć wartość zero. Rejestr IDTR ładuje się w programie rozkazem *LIDT*

Jak już wspomniano, mikroprocesor 80286 może pracować w *trybie adresowania rzeczywistego* lub *wirtualnego* z ochroną zasobów, zależnie od wartości najmniej znaczącego bitu **PE** (ang. *protected enable*) w rejestrze *słowa stanu maszyny* **MSW** (ang. *machine status word*). Format słowa MSW przedstawiono na rys. 2.48.



Rys. 2.48. Format słowa stanu maszyny MSW

Wykorzystywane bity słowa stanu maszyny mają następujące znaczenie:

b_0 = **PE** = 1 (ang. *protected mode enable*) ustawia mikroprocesor 80286 w tryb adresowania wirtualnego z ochroną zasobów. Bit ten można wyzerować tylko zewnętrznym sygnałem zerującym (RESET = 1), a ustawić rozkazem *LMSW* w programie;

b_1 = **MP** (ang. *monitor processor extension*) – znacznik obecności koprocesora. Rozkazem *WAIT*, przy braku koprocesora (MP = 0), można pobudzić przerwanie wewnętrzne o wektorze adresu przerwań $n = 7$;

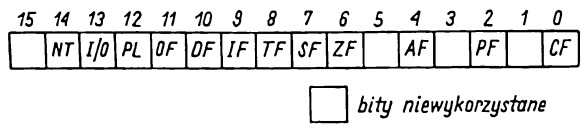
b_2 = **EM** (ang. *emulate processor extension*) – znacznik emulacji. Przy braku koprocesora (EM = 1), rozkazem *ESC* można pobudzić przerwanie wewnętrzne, uruchamiające procedurę emulacji koprocesora od adresu określonego wektorem przerwania o numerze $n = 7$;

b_3 = **TS** (ang. *task switched*) – znacznik, ustawiany w stan 1 po każdej operacji przełączania zadania wskazuje, że stan koprocesora może należeć do zadania różnego od bieżącego.

Po zerowaniu wszystkie bity rejestru MSW są wyzerowane i mikroprocesor 80286 działa tak, jak mikroprocesor 8086.

Przedstawiony na rysunku 2.49 rejestr znaczników **F** mikroprocesora 80286 zawiera znaczniki stanu (AF, CF, OF, SF, PF, ZF) i znaczniki sterujące (DF, IF, TF) identyczne ze znacznikami rejestru **F** mikroprocesorów 8086/88, a ponadto dwa specjalne pola **I/OPL** i **NT**. Dwubitowe pole I/OPL (ang. *input output privilege level*) określa poziom uprzywilejowania układów we–wy. Znacznik **NT** (ang. *nested task*) wskazuje zagnieżdżanie zadań. Steruje on przebiegiem rozkazu *IRET*. Gdy NT = 0, rozkaz powrotu nie zmienia bieżącego zadania (segmentu), a gdy NT = 1, następuje przełączenie zadania (powrót do zadania poprzedniego sprzed przerwania).

Znacznik NT jest ustawiany rozkazem *CALL* lub przerwaniem INT, inicjującym przełączanie zadania. Przerwaniem INT, nie inicjującym przełączania zadania, zeruje się znacznik NT. Znacznik NT można także ustawiać lub zerować rozkazami *POPF* lub *IRET* (patrz p. 4.2).



Rys. 2.49. Rejestr znaczników F mikroprocesora 80286

Sposoby adresowania argumentów w pamięci

Mikroprocesor 80286 (podobnie jak mikroprocesory 8086/88) ma 8 głównych trybów adresowania argumentu:
- *rejestrowe bezpośrednie* (argument znajduje się w rejestrze 8 lub 16-bitowym wskazanym przez rozkaz);
- *natychmiastowe* (argument jest zawarty w rozkazie);
- *rejestrowe pośrednie* (położenie (offset) określa się zawartością jednego z rejestrów SI, DI, BX);
- *bezpośrednie* (położenie jest zawarte w rozkazie jako 8 lub 16-bitowe przesunięcie);
- *bazowe* (położenie jest sumą 8 lub 16-bitowego przesunięcia zawartego w rozkazie i zawartości jednego z rejestrów bazowych BP lub BX);
- *indeksowe* (położenie jest sumą 8 lub 16-bitowego przesunięcia zawartego w rozkazie i zawartości jednego z rejestrów indeksowych SI lub DI);
- *bazowo-indeksowe* (położenie jest sumą zawartości jednego z rejestrów indeksowych i jednego z rejestrów bazowych);
- *bazowo-indeksowe z przesunięciem* (położenie jest sumą zawartości rejestru bazowego, indeksowego i przesunięcia 8 lub 16-bitowego zawartego w rozkazie).

Adresowanie układów we-wy

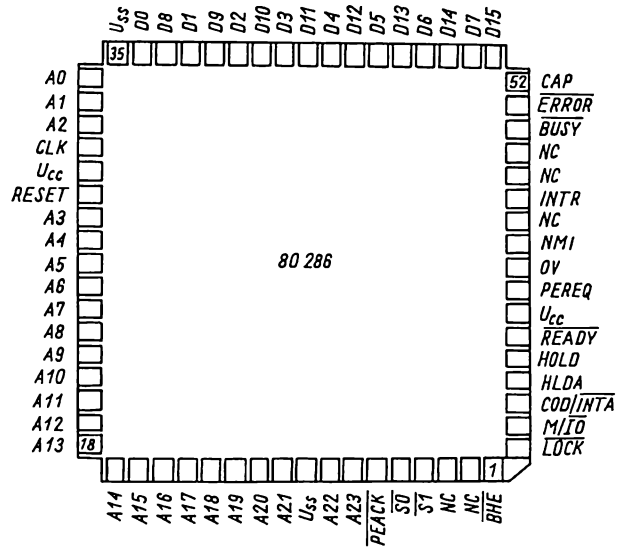
W mikroprocesorze 80286 układy we-wy mogą być adresowane w jednolitej bądź rozdzielonej przestrzeni adresowej. W pierwszym przypadku układy we-wy są traktowane jak komórki pamięci, w drugim – mikroprocesor 80286 (podobnie jak 8086) dysponuje 64 kilobajtową przestrzenią adresową układów we-wy (można zaadresować 64 kilo 8-bitowych lub 32 kilo 16-bitowych portów we-wy). Przestrzeń ta nie jest segmentowana. Porty we-wy można adresować bezpośrednio (dotyczy to rozkazów we-wy, w których po kodzie operacji występuje 8-bitowy adres portu) lub pośrednio rejestrowo (dotyczy to rozkazów we-wy, których kod operacji wskazuje rejestr DX, zawierający 16-bitowy adres portu w przestrzeni we-wy od 0000 do FFFFH. Obszar we-wy o adresach od 00F8H do 00FFH zarezerwowano dla potrzeb mikroprocesora (który używa adresów F8, FA, FC przy wykonywaniu rozkazów ESC, gdy EM = 0 i TS = 0) i systemu operacyjnego. Na rysunku 2.50 przedstawiono rozmieszczenie wyprowadzeń mikroprocesora 80286.

Przeznaczenie poszczególnych wyprowadzeń jest następujące:

D15...D0 – dwukierunkowa trójstanowa magistrala danych;

A23...A0 – jednokierunkowa trójstanowa magistrala adresowa;

$\overline{\text{BHE}}$ – wyjściowy sygnał trójstanowy, umożliwiający łącznie z linią A0, podobnie jak w mikroprocesorze 8086, przesłania 8 lub 16-bitowe (patrz tabl. 2.5);



Rys. 2.50. Rozmieszczenie wyprowadzeń mikroprocesora 80286

Tablica 2.9

COD/$\overline{\text{INTA}}$	M/$\overline{\text{IO}}$	$\overline{\text{S1}}$	$\overline{\text{S0}}$	Rodzaj cyklu magistrali
0	0	0	0	Cykl potwierdzenia przerwania
0	0	0	1	Nie występuje
0	0	1	0	Nie występuje
0	0	1	1	Stan „pasywny" statusu
0	1	0	0	Jeżeli A1 = 1, to cykl zatrzymania (ang. *halt*), w przeciwnym razie cykl wstrzymania (ang. *shutdown*).
0	1	0	1	Cykl odczytu danych z pamięci
0	1	1	0	Cykl zapisu danych do pamięci
0	1	1	1	Stan „pasywny" statusu
1	0	0	0	Nie występuje
1	0	0	1	Cykl odczytu we-wy
1	0	1	0	Cykl zapisu we-wy
1	0	1	1	Stan „pasywny" statusu
1	1	0	0	Nie występuje
1	1	0	1	Cykl pobrania kodu
1	1	1	0	Nie występuje
1	1	1	1	Stan „pasywny" statusu

$\overline{\text{S1}}$, $\overline{\text{S0}}$ – wyjściowe sygnały trójstanowe statusu mikroprocesora, określające, wspólnie z sygnałami M/$\overline{\text{IO}}$ i COD/$\overline{\text{INTA}}$, rodzaj cyklu magistrali. Magistrala jest w tzw. *cyklu statusu* T_s w przypadku, gdy jeden lub dwa sygnały $\overline{\text{S1}}$, $\overline{\text{S0}}$ są aktywne. Rodzaje cykli magistrali przedstawiono w tabl. 2.9. Do wy-

prowadzeń tych należy doprowadzić napięcie +5 V przez zewnętrzne rezystory 20 kΩ ± 10%, w celu utrzymania ich w stanie nieaktywnym poczas *stanu zawieszenia* (HOLD) mikroprocesora;

M/$\overline{IO}$ – wyjściowy sygnał trójstanowy, wskazujący podczas cyklu T_s stanem wysokim M/$\overline{IO}$ = 1 cykl pamięci lub cykl zatrzymania (ang. *halt*) bądź wstrzymania (ang. *shutdown*), a stanem niskim M/$\overline{IO}$ = 0 – cykl we–wy lub potwierdzenia przerwania maskowalnego;

COD/$\overline{INTA}$ – wyjściowy sygnał trójstanowy, umożliwiający rozróżnienie cyklu magistrali pobrania rozkazu od cyklu odczytu danych oraz cyklu potwierdzenia przerwania od cyklu we–wy. Przebiegi czasowe sygnałów M/$\overline{IO}$ i COD/$\overline{INTA}$ są jednakowe;

$\overline{LOCK}$ – wyjściowy sygnał trójstanowy, wskazujący stanem niskim ($\overline{LOCK}$ = 0), że inny moduł nadrzędny nie może przejąć sterowania magistrali. Sygnał ten ustawia się na 0, umieszczając w programie przedrostek LOCK przed rozkazem ($\overline{LOCK}$ = 0 podczas wykonywania tego rozkazu) lub automatycznie za pośrednictwem układu sterowania mikroprocesora podczas wykonywania rozkazu *XCHG*, cyklu potwierdzenia przerwania oraz dostępu do tablicy deskryptorów;

$\overline{READY}$ – wejściowy sygnał, wskazujący stanem niskim (READY = 0) zakończenie cyklu magistrali, w przeciwnym przypadku – potrzebę wydłużenia cyklu magistrali o takty oczekiwania T_w. W stanie zawieszenia HOLD syganł READY jest ignorowany. Do wyprowadzenia READY należy doprowadzić napięcie +5 V przez rezystor 910 Ω ± 5%, aby zapewnić przejście sygnału $\overline{READY}$ do stanu nieaktywnego, w minimalnym wymaganym czasie;

HOLD – wejściowy sygnał żądania zwolnienia magistrali na rzecz innego modułu nadrzędnego;

HLDA – wyjściowy sygnał, potwierdzający przejście mikroprocesora do stanu zawieszenia (HOLD), sygnalizujący zwolnienie magistrali na rzecz modułu nadrzędnego generującego sygnał HOLD;

INTR – wejściowy sygnał żądania przerwania maskowalnego, aktywny poziomem wysokim. Poziom aktywny powinien być utrzymywany do czasu zakończenia pierwszego cyklu magistrali w cyklu potwierdzenia przerwania (cykl potwierdzenia przerwania zawiera 2 cykle magistrali niezbędne do pobrania 8-bitowego wektora przerwań – patrz p. 2.1). Sygnał ten jest testowany na początku każdego cyklu magistrali i musi być uaktywniony przynajmniej na dwa takty procesora przed zakończeniem rozkazu bieżącego, gdyż w przeciwnym razie przerwanie będzie przyjęte dopiero po zakończeniu następnego rozkazu;

NMI – wejściowy sygnał żądania przerwania niemaskowalnego. Przerwanie to, podobnie jak w mikroprocesorze 8086, jest obsługiwane procedurą rozpoczynającą się od adresu określonego wektorem przerwania $n = 2$. W celu poprawnego przyjęcia przerwania niemaskowalnego, sygnał NMI powinien być utrzymywany w stanie aktywnym (wysokim) przez co najmniej 4 takty zegarowe, przy czym odstęp między dwoma kolejnymi aktywnymi sygnałami NMI musi być większy od 4 taktów zegarowych;

PEREQ – (ang. *processor extension operand request*) – sygnał wejściowy sygnalizujący mikroprocesorowi poziomem wysokim, że z koprocesora wysłano żądanie przesłania argumentu (danych);

$\overline{\text{PEACK}}$ – (ang. *processor extension operand acknowledge*) – sygnał wyjściowy sygnalizujący koprocesorowi, że odbywa się przesyłanie argumentu. Sygnałami PEREQ i $\overline{\text{PEACK}}$ rozszerza się możliwości mikroprocesora 80286 w zakresie zarządzania i ochrony pamięci na koprocesory. Do wyprowadzenia PEACK należy doprowadzić napięcie o wartości +5 V przez rezystor o wartości 20 kΩ ± 10%, w celu utrzymania stanu nieaktywnego sygnału $\overline{\text{PEACK}}$ podczas stanu zawieszenia (HOLD) mikroprocesora;

$\overline{\text{BUSY}}$ – (ang. *processor extension busy*) – sygnał wejściowy zatrzymujący stanem aktywnym (0) wykonywanie programu na rozkazie *WAIT* i niektórych rozkazach *ESC* do czasu, aż sygnał ten stanie się nieaktywny. Mikroprocesor 80286 może przyjmować przerwania podczas oczekiwania na zmianę stanu sygnału $\overline{\text{BUSY}}$ z 0 na 1;

$\overline{\text{ERROR}}$ – (ang. *processor extension error*) – sygnał wejściowy, który w stanie aktywnym pobudza mikroprocesor do wygenerowania przerwania od koprocesora podczas wykonywania rozkazu *WAIT* lub niektórych rozkazów *ESC*. Do wyprowadzeń $\overline{\text{BUSY}}$ i $\overline{\text{ERROR}}$ należy doprowadzić napięcie o wartości +5 V przez rezystor o wartości 20 kΩ ± 10%, w celu utrzymania tych sygnałów w stanie nieaktywnym przy braku koprocesora 80287;

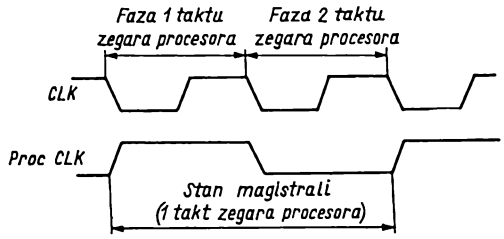
RESET – wyjściowy sygnał zerowania systemowego, aktywny stanem wysokim. Podczas zerowania wyprowadzenia mikroprocesora są w następujących stanach:
– $\overline{\text{S0}}$, $\overline{\text{S1}}$, $\overline{\text{PEACK}}$, A23...A0, $\overline{\text{BHE}}$, $\overline{\text{LOCK}}$ – w stanie 1,
– M/$\overline{\text{IO}}$, COD/$\overline{\text{INTA}}$, HLDA – w stanie 0,
– D15...D0 – w stanie dużej impedancji;

CLK – wejściowy sygnał systemowych impulsów zegarowych, synchronizujących pracę mikroprocesora 80286 (po podzieleniu ich częstotliwości przez 2 w wewnętrznym dzielniku);

U_{CC}, U_{SS} – wyprowadzenia napięć zasilania +5 V, 0 V;

CAP – (ang. *substrate filter capacitor*) – wejście służące do dołączenia pojemnościowego filtra (zwykle kondensator o pojemności – 0,0047 μF ±20%, napięciu – 12 V, dołączony między to wyprowadzenie a masę).

W mikroprocesorze 80286 (patrz tablica 2.9) można wyróżnić 7 *cykli magistrali*. Są one synchronizowane impulsami zegarowymi CLK (z układu 82284 patrz rys. 2.55). Wewnątrz mikroprocesora z impulsów zegarowych CLK są tworzone impulsy synchronizujące mikroprocesora ProcCLK o częstotliwości dwa razy mniejszej niż częstotliwość impulsów CLK. Jeden takt T_p mikroprocesora zawiera 2 takty T zegara systemowego – rys. 2.51. Takt T_p mikroprocesora określa czas trwania stanu magistrali.



Rys. 2.51. Przebiegi czasowe impulsów zegarowych CLK i ProcCLK

W mikroprocesorze 80286 występują 4 różne stany (ang. *states*) magistrali:

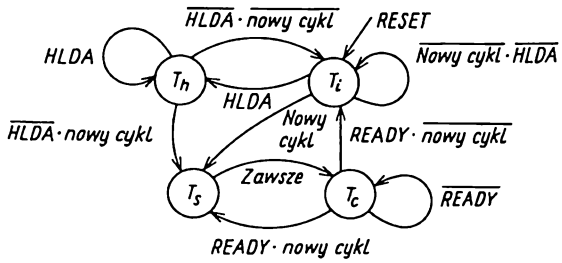
T_h – zawieszenia (ang. *hold*)
T_i – bezczynności (ang. *idle*)
T_s – statusu (ang. *status*)
T_c – rozkazu (ang. *command*)

Stan zawieszenia T_h jest osiągany po podaniu sygnału HOLD = 1. W stanie tym linie adresowe, danych i wyjściowych sygnałów sterujących, w tym i statusu, są ustawiane w stan dużej impedancji. Mikroprocesor 80286, sygnałem HLDA = 1, sygnalizuje innym modułom nadrzędnym przejście magistrali do stanu T_h.

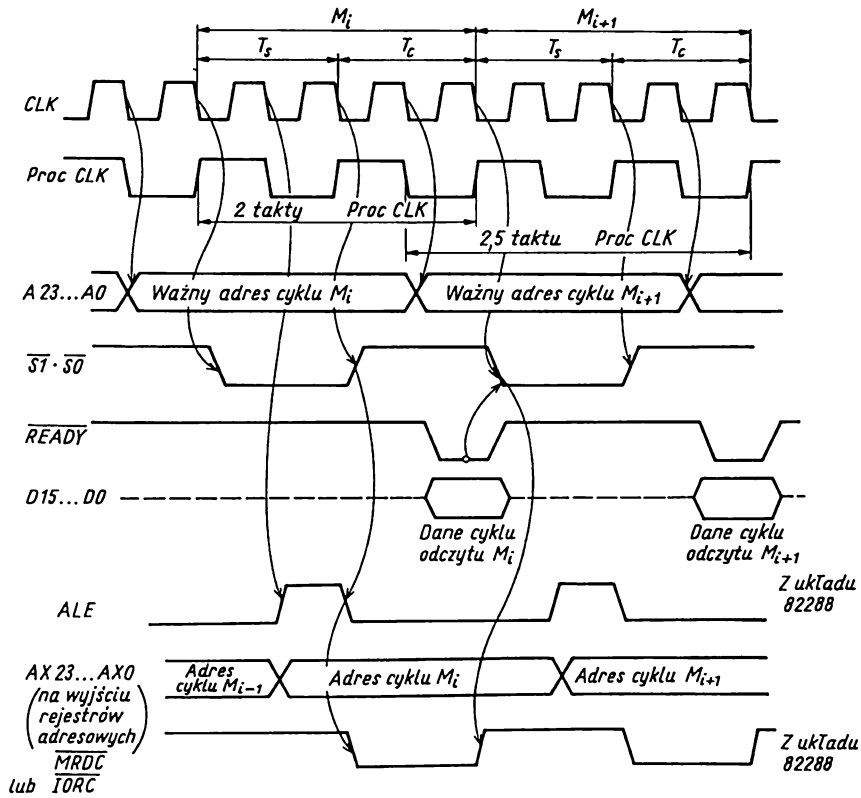
Stan bezczynności T_i oznacza, że magistrala nie jest wykorzystywana (nie odbywa się przesyłanie danych i żaden z modułów nadrzędnych nie żąda dostępu do magistrali).

Stan statusu T_s jest osiągany w wyniku pojawienia się aktywnego statusu $\overline{S1}\,\overline{S0}$, tzn. $\overline{S1}\,\overline{S0}$ = 01, $\overline{S1}\,\overline{S0}$ = 10 lub $\overline{S1}\,\overline{S0}$ = 00. W tym stanie następuje dekodowanie rozkazu oraz ustawienie ważnego adresu i danych (dla cykli zapisu). Sterownik magistrali 82288 (patrz rys. 2.57) dekoduje sygnały statusu i generuje, odpowiednie do realizowanego cyklu magistrali, sygnały sterujące.

Po stanie T_s magistrala procesora automatycznie przechodzi do *stanu rozkazu* T_c, w którym następuje odczytanie bądź zapisanie pamięci lub układu we–wy. Stan ten można wydłużyć o wymaganą liczbę taktów oczekiwania T_w (w tym przypadku kolejnych stanów T_c), zależną od czasu dostępu pamięci lub układów we–wy. Wydłużenie stanu T_c następuje w przypadku, gdy sygnał $\overline{READY}$ = 1. Przejścia między stanami magistrali mikroprocesora 80286 przedstawiono w postaci grafu na rys. 2.52.



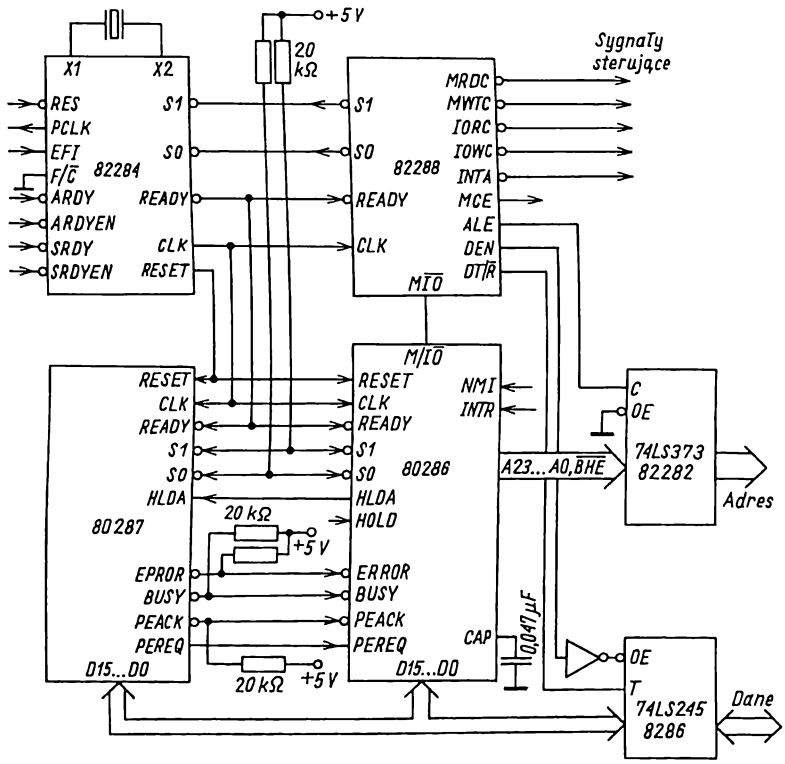
Rys. 2.52. Graf przejść dla stanów magistrali mikroprocesora 80286



Rys. 2.53. Podstawowe cykle magistrali

Cykl magistrali mikroprocesora 80286 wymaga 2,5 taktu (stanu) mikroprocesora (5 taktów zegarowych CLK z tym, że dzięki strukturze potokowej ostatni takt CLK i-tego cyklu magistrali zachodzi na pierwszy takt CLK (i + 1)-cyklu magistrali

(w tym takcie odbywa się przesłanie danych i-tego cyklu magistrali). Z przedstawionego na rys. 2.53 przykładu dwóch kolejnych cykli odczytu wynika, że zbyt krótkie utrzymywanie adresu w mikroprocesorze 80286 nie zapewnia właściwego przebiegu cyklu odczytu lub zapisu. W zasadzie gwarantowane jest utrzymywanie adresu i-tego cyklu tylko podczas pierwszej fazy pierwszego stanu T_c, dlatego też pojawiający się w cyklu odczytu lub zapisu adres musi być zapamiętywany w rejestrze zewnętrznym, sterowanym sygnałem ALE ze sterownika magistrali 82288. Dzięki temu uzyskuje się czas dostępu (od adresu do ważnych danych odczytywanych) równy 2,5 taktu mikroprocesora przy czasie cyklu magistrali równym 2 takty mikroprocesora. Adres dla (i+1)-cyklu magistrali pojawia się podczas drugiej fazy ostatniego stanu T_c i-tego cyklu magistrali.

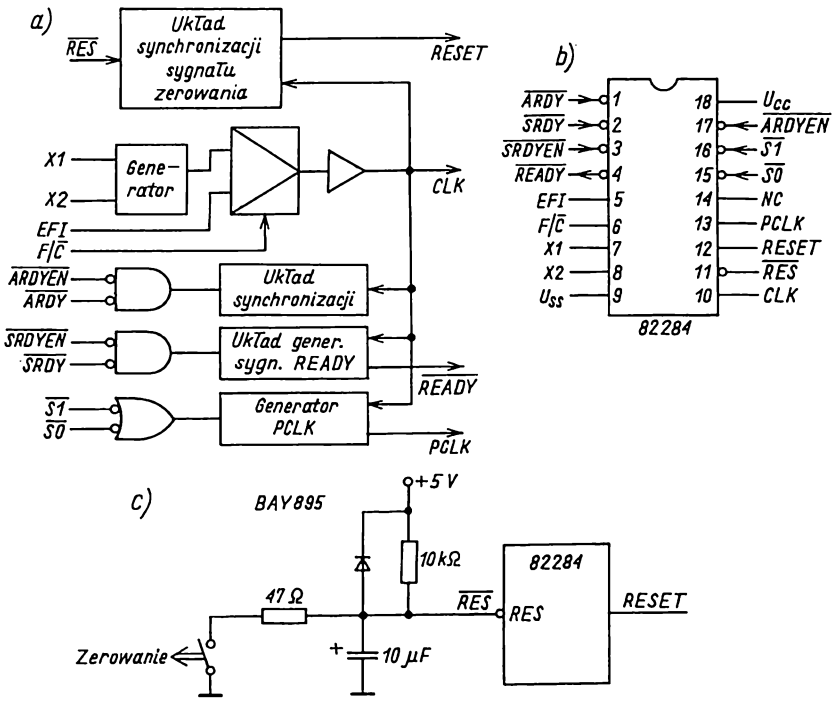


Rys. 2.54. Jednostka centralna z mikroprocesorem 80286

Jednostka centralna oprócz mikroprocesora 80286 zawiera układ zegara systemowego 82284, układ sterownika magistrali 82288, rejestry adresowe i bufory danych – rys. 2.54. Układ zegara 82284 (rys. 2.55a, b) generuje wymagane sygnały zegarowe CLK przesyłane do mikroprocesora i PCLK – do układów we-wy, formuje i synchronizuje sygnał zerowania RESET oraz generuje sygnał $\overline{\text{READY}}$ na podstawie wejścio-

wego sygnału synchronicznego $\overline{SRDY}$ lub asynchronicznego $\overline{ARDY}$, używanego do określania niskim poziomem końca bieżącego cyklu magistrali. Sygnał zerowania RESET powinien być utrzymywany na wysokim poziomie przez co najmniej 16 taktów CLK, co zapewnia układ przedstawiony na rys. 2.55c, zarówno po naciśnięciu przycisku zerującego, jak i po włączeniu napięcia zasilania. Sygnał zerowania wprowadza mikroprocesor 80286 w tryb adresowania rzeczywistego, przenosi sterowanie do komórki o adresie fizycznym FFFFF0H i ustawia w wybranych rejestrach mikroprocesora 80286 następujące wartości początkowe:

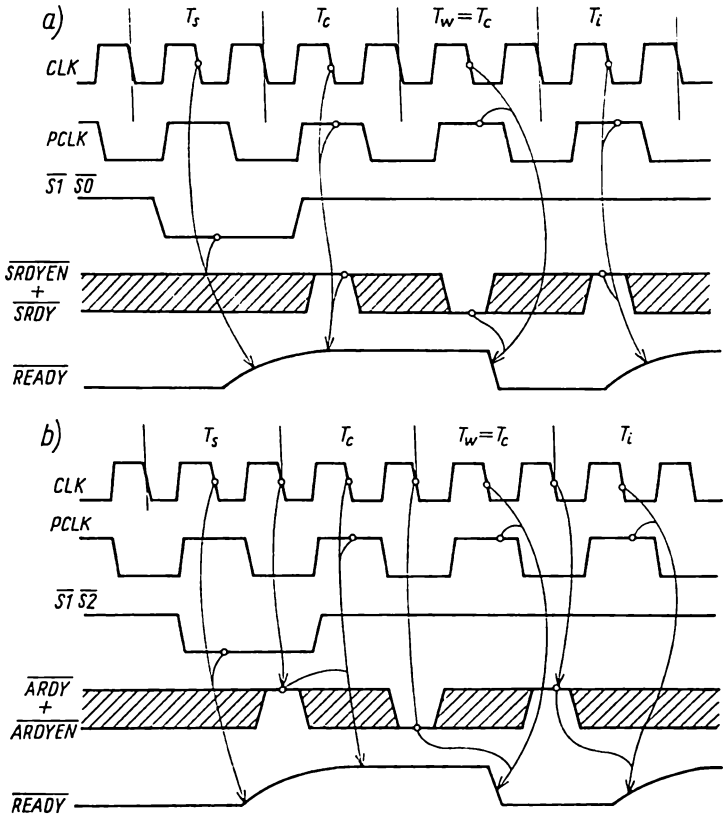
F ← 0002H
MSW ← FFF0H
IP ← FFF0H
CS ← F000H
DS ← 0000H
ES ← 0000H
SS ← 0000H



Rys. 2.55. Układ zegara 82284: a) schemat blokowy; b) rozmieszczenie wyprowadzeń; c) dołączenie sygnału $\overline{RES}$

Po zerowaniu mikroprocesor może przejść do stanu zawieszenia (przy HOLD = 1) dopiero po 34 taktach CLK od chwili zaniku sygnału RESET.

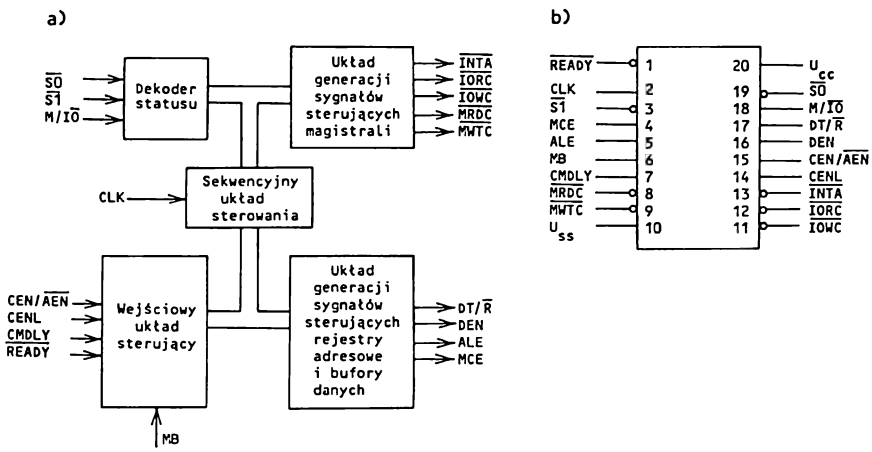
Sygnały gotowości $\overline{SRDY}$, $\overline{ARDY}$ są blokowane odpowiednio sygnałami $\overline{SRDYEN}$ i $\overline{ARDYEN}$. Zwykle dekoder adresów wybiera jedno z tych wejść blokujących. Sygnał $\overline{READY}$ pojawia się, jeżeli $\overline{SRDY} + \overline{SRDYEN} = 0$ lub $\overline{ARDY} + \overline{ARDYEN} = 0$ i pozostaje aktywny przez dwa takty CLK.



Rys. 2.56. Sposób generowania sygnału $\overline{READY} = 0$ za pomocą wejść: a) synchronicznych; b) asynchronicznych

Przebiegi czasowe przy generowaniu sygnału $\overline{READY} = 0$ pod wpływem $\overline{SRDY} = 0$ przedstawiono na rys. 2.56a, a pod wpływem $\overline{ARDY} = 0$ – na rys. 2.56b. W pierwszym przypadku stan sumy sygnałów $\overline{SRDY} + \overline{SRDYEN}$ jest próbkowany na opadającym zboczu CLK przy pasywnym statusie ($\overline{S1}\,\overline{S0} = 11$) i sygnale PCLK = 1. Przy $\overline{SRDY} + \overline{SRDYEN} = 1$ cykl magistrali jest wydłużony o stan $T_c = T_w$, a przy $\overline{SRDY} + \overline{SRDYEN} = 0$ cykl magistrali jest kończony. W drugim przypadku stan sumy sygnałów $\overline{ARDY} + \overline{ARDYEN}$ jest próbkowany przez wewnętrzny układ synchronizatora, na każdym opadającym zboczu impulsu CLK, następnie sygnał z synchronizatora jest próbkowany przy sygnale PCLK = 1. Gdy $\overline{ARDY} + \overline{ARDYEN} = 0$,

sygnały na wejściach synchronicznych SRDY i SRDYEN są ignorowane. Suma $\overline{ARDY}$ + $\overline{ARDYEN}$ powinna być równa 1 pod koniec stanu T_s.



Rys. 2.57. Sterownik magistrali 82288: a) schemat blokowy; b) rozmieszczenie wyprowadzeń

Sterownik magistrali 82288 (rys. 2.57), na podstawie sygnałów M/$\overline{IO}$, $\overline{S1}$, $\overline{S0}$ generuje sygnały sterujące, niezbędne do zapewnienia prawidłowego przebiegu zdekodowanego cyklu magistrali oraz sterowania rejestrów adresowych i buforów magistrali danych. Może on pracować w dwóch trybach:

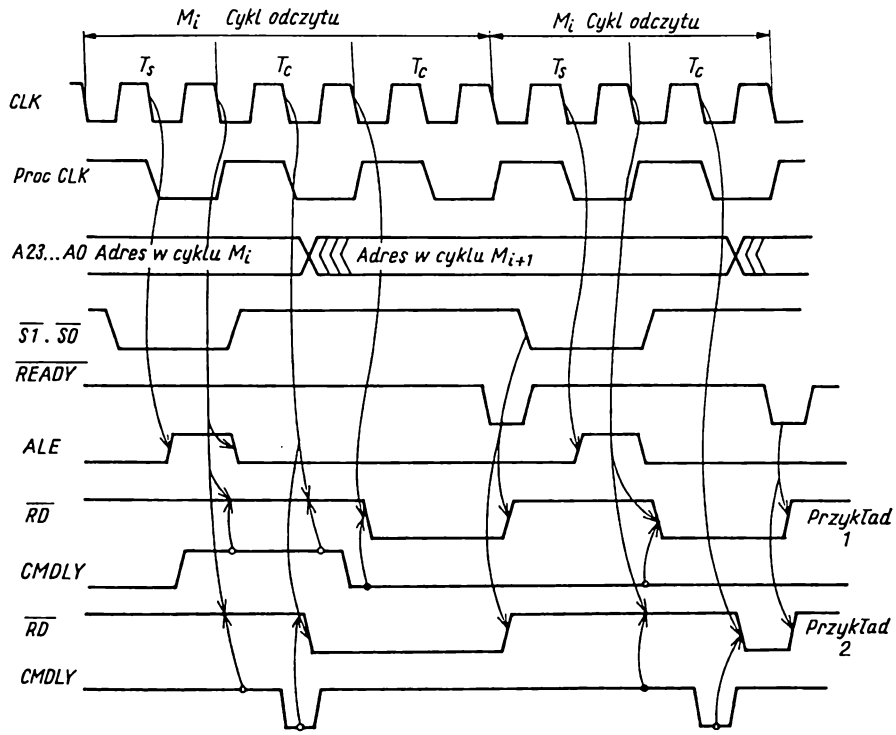
- w pierwszym umożliwia realizację cykli magistrali zgodnie z wymaganiami magistrali Multibus (IEEE–796), w których narzucono konieczność wtrącania co najmniej jednego taktu T_w, ze względu na opóźnienie sygnałów sterujących cyklu magistrali;
- w drugim umożliwia realizację cykli magistrali z dużą szybkością (bez wtrącenia taktów oczekiwania T_w).

Pierwszy tryb jest wybierany ustawieniem stanu sygnału MB (ang. *multibus mode select*) na 1, a drugi – ustawieniem stanu sygnału MB na 0. Sygnał MB określa również funkcje sygnału wejściowego CEN/$\overline{AEN}$ sterownika magistrali.

Sygnał CEN/$\overline{AEN}$ (ang. *command enable/address enable*) steruje trójstanowe wyjścia sterujące cyklu magistrali $\overline{INTA}$, $\overline{IORC}$, $\overline{IOWC}$, $\overline{MRDC}$, $\overline{MWTC}$ (ich przeznaczenie jest identyczne do określonego w sterowniku 8288 – patrz rys. 2.18) i wyjście DEN. Stan sygnału CEN/$\overline{AEN}$ zwykle ustawia się na 1 lub 0. Przy MB = 1 wybrana jest funkcja $\overline{AEN}$. $\overline{AEN}$ = 0 sygnalizuje zwolnienie magistrali przez inne urządzenie nadrzędne i sterownik może dołączyć ją do mikroprocesora. Sygnały sterujące cyklu magistrali mogą wyjść ze stanu dużej impedancji i przejść do stanu nieaktywnego. Sygnał $\overline{AEN}$ = 1 oznacza, że inny moduł nadrzędny steruje magistralą i sterownik magistrali powinien odłączyć ją od mikroprocesora (sygnały sterujące cyklu magistrali przechodzą do stanu dużej impedancji, a sygnał DEN – do stanu nieaktywnego).

Przy MB = 0 jest wybrana funkcja CEN. CEN = 1 uaktywnia sygnały sterujące cyklu magistrali i DEN, a CEN = 0 wprowadza je do stanu nieaktywnego (nie są one jednak wprowadzane do stanu dużej impedancji).

Sygnał wejściowy CMDLY (ang. *command delay*) umożliwia opóźnienie pojawienia się odpowiednich sygnałów sterujących cyklu magistrali (patrz tab. 2.9). Jeżeli sygnał CMDLY = 1 (na opadającym zboczu CLK kończącym stan T_S), to sygnały sterujące cyklu magistrali nie będą uaktywnione aż do chwili pojawienia się sygnału CMDLY = 0 (następne próbkowania sygnału CMDLY odbywają się każdym kolejnym opadającym zboczem CLK), przy czym pojawienie się sygnału READY = 0 przed uaktywnieniem sygnałów sterujących cyklu magistrali spowoduje zakończenie tego cyklu. Wpływ stanu sygnału CMDLY na przebieg cyklu magistrali przedstawiono na rys. 2.58.

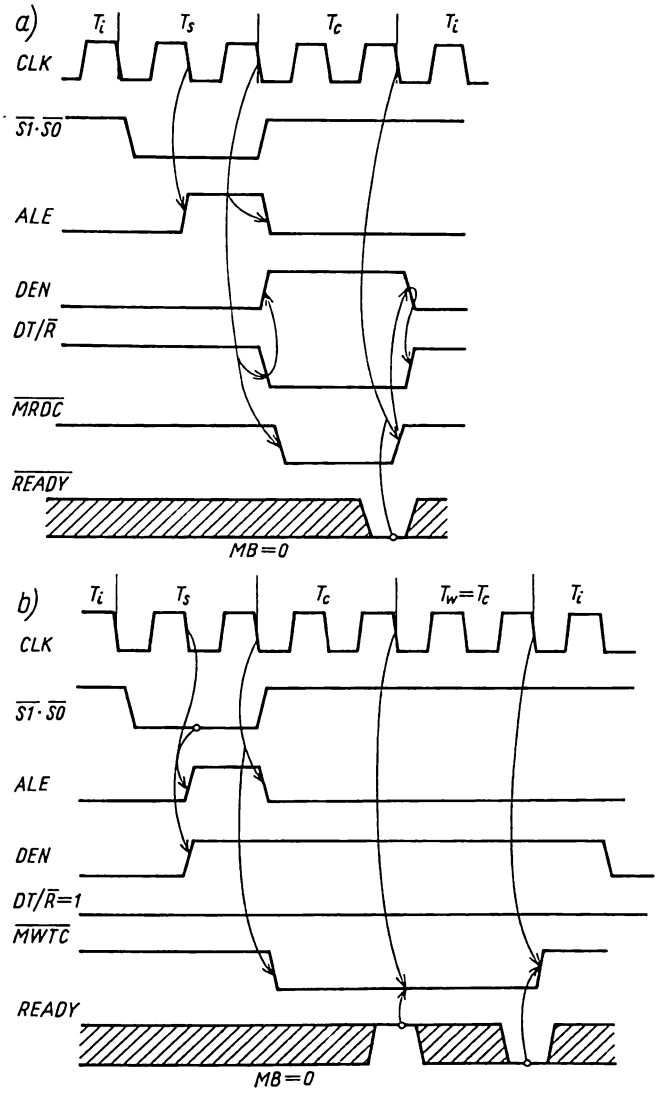


Rys. 2.58. Wpływ sygnału CMDLY na przebieg cyklu magistrali

Pozostałe sygnały mają jednakowe funkcje z odpowiednikami w mikroprocesorze 8086 i sterowniku magistrali 8288, przy czym sygnał CENL (ang. *command enable latched*) jest używany do wyboru (uaktywniania) odpowiedniego sterownika magistrali. W systemie wielomagistralowym sygnał CENL jest zapisywany w wewnętrznym rejestrze (typu zatrzask) na końcu każdego stanu T_S.

Przebiegi czasowe cykli odczytu i zapisu przy sygnale MB = 0 przedstawiono na rys. 2.59, a cykli odczytu i zapisu przy MB = 1 – na rys. 2.60. Odpowiednie sygnały sterujące cyklu magistrali są uaktywniane opadającym zboczem sygnału CLK kończą-

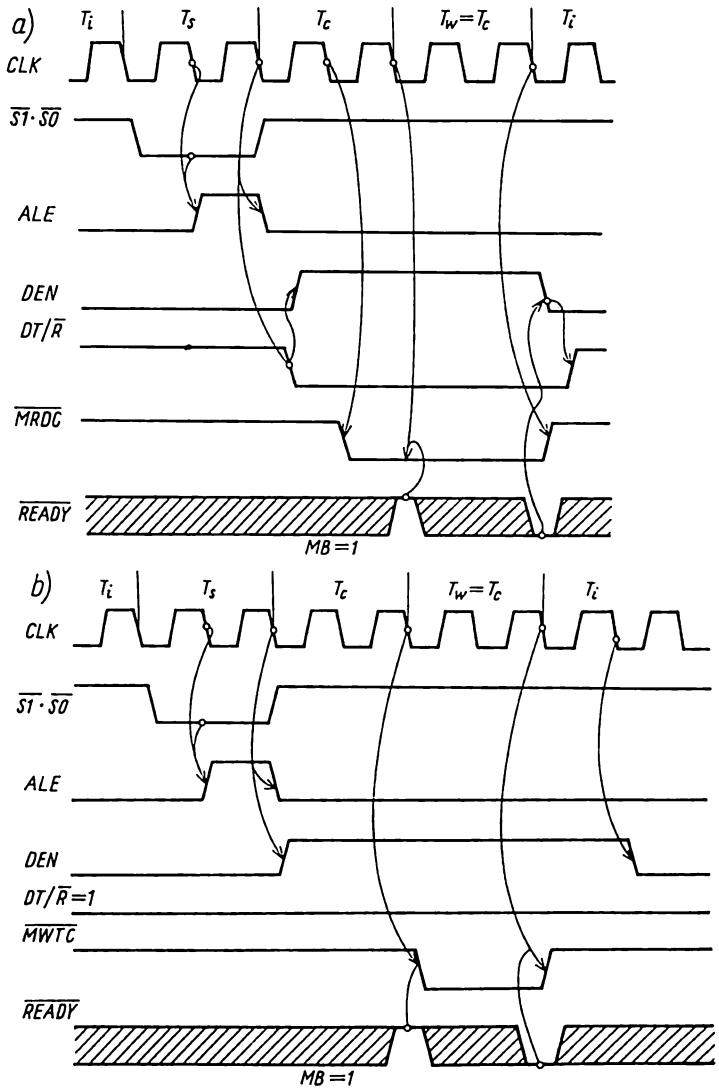
cego stan T_s, przy MB = 0 lub opadającym zboczem sygnału CLK w połowie stanu T_c ($\overline{MRDC}$) i na końcu stanu T_c ($\overline{MWTC}$), przy MB = 1 (ze względu na to opóźnienie jest wymagany przynajmniej jeden takt T_w).



Rys. 2.59. Przebiegi czasowe cykli odczytu i zapisu, gdy sygnał MB = 0

Układ przerwań w mikroprocesorze 80286 rozwiązano podobnie, jak w mikroprocesorze 8086/88, z tym, że pojawiają się dodatkowe przerwania wewnętrzne

wywoływane pewnymi wyjątkowymi zdarzeniami (ang. *exception*) w trybie adresowania rzeczywistego i wirtualnego, prowadzącymi do błędnej realizacji rozkazu (błędnej pracy systemu).

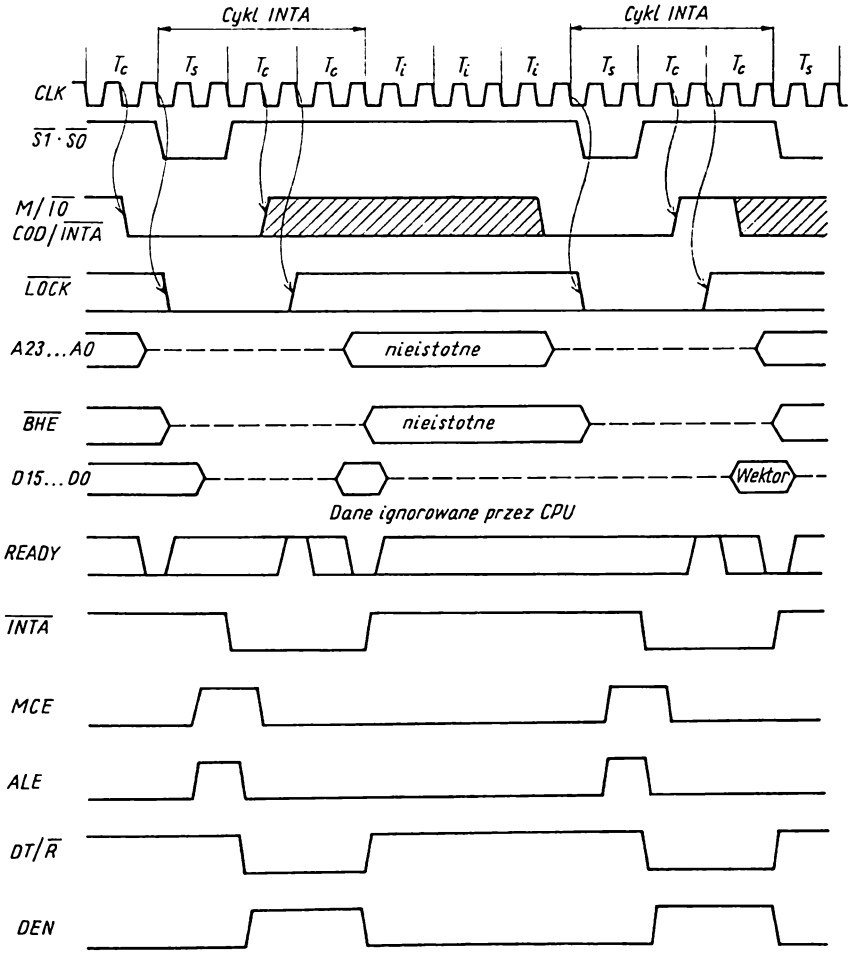


Rys. 2.60. Przebiegi czasowe cykli odczytu i zapisu, gdy sygnał MB = 1

Tym dodatkowym przerwaniom wewnętrznym przyporządkowano stałe wektory adresów przerwań o numerach od 5 do 16 (poziomy przerwań od 5 do 16).

Przyjęcie takiego przerwania, podobnie jak i pozostałych przerwań wewnętrznych oraz zewnętrznego *przerwania niemaskowalnego* NMI nie wymaga realizacji cyklu potwierdzenia, gdyż przyporządkowano im stałe adresy początkowe obsługujących je podprogramów.

Przyporządkowanie wektorów adresów przerwań (poziomów przerwań) poszczególnym przerwaniom przedstawiono w tablicy w rozdziale czwartym. *Przerwanie maskowalne* INTR może być przyjęte, jeżeli bit IF w rejestrze F jest ustawiony w stan 1 i nie ma innego przerwania o wyższym priorytecie.

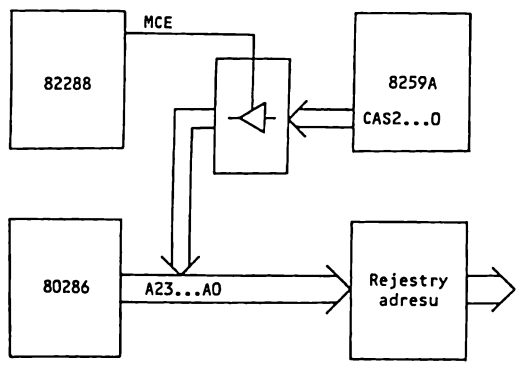


Rys. 2.61. Przebiegi czasowe cyklu potwierdzenia przerwania

W cyklu *potwierdzenia przerwania* (rys. 2.61) mikroprocesor czeka na 1-bajtowy wektor adresu przerwania wysyłany przez sterownik 8259A. Może to być

jeden z 224 wektorów adresów przerwań (od 32 do 255), definiowanych przez użytkownika. Oczywiście użytkownik może tak zaprogramować sterownik przerwań, aby były w nim generowane również wektory adresów przerwania od 00H do 31H, ale byłoby to nierozsądne (gdyż są im przyporządkowane procedury obsługi określonych przerwań wewnętrznych). Cykl potwierdzenia przerwania składa się z dwóch cykli magistrali.

W pierwszym cyklu INTA, przy kaskadowej strukturze systemu przerwań, nadrzędny sterownik magistrali liniami CAS0...2 wskazuje, który ze sterowników podrzędnych, podczas drugiego cyklu INTA, ma wysłać wektor adresu, określajacy adres startowy procedury obsługi przyjętego przerwania. Sygnał MCE (wytwarzany w układzie 82288), przy kaskadowej strukturze przerwań, jest stosowany w cyklu potwierdzenia przerwania do odblokowania trójstanowych buforów, umożliwiających dołączenie wyjść adresowych CAS0...CAS2 nadrzędnego sterownika przerwań do lokalnej magistrali adresowej mikroprocesora. Generowany na tych wyjściach adres jest przekazywany do podrzędnych sterowników przerwań systemową magistralą adresową przez rejestry adresowe – rys. 2.62, aby wskazać ten z nich, który powinien wygenerować wektor adresu przerwania. Podczas cykli potwierdzania mikroprocesor generuje sygnał $\overline{LOCK}$ = 0, zapobiegający zwolnieniu magistrali systemowej między tymi cyklami dla innego modułu nadrzędnego. Lokalne żądanie przejścia mikroprocesora do stanu zawieszenia (sygnał HOLD = 1) jest ignorowane do czasu zakończenia drugiego cyklu potwierdzenia przerwania.



Rys. 2.62. Sposób wykorzystania sygnału MCE

Aby zapewnić wymagany w sterowniku 8259A odstęp między sygnałami $\overline{INTA}$, mikroprocesor 80286 "wtrąca" 3 takty (stany) bezczynności między pierwszy i drugi cykl potwierdzenia przerwania. Sygnałem gotowości $\overline{READY}$ należy wydłużyć oba cykle potwierdzenia przerwania o co najmniej jeden takt oczekiwania T_w, w celu zapewnienia minimalnej długości impulsu $\overline{INTA}$ = 0. Przy jednoczesnym wystąpieniu kilku różnych typów przerwań będą one przyjmowane w następującej kolejności:

przerwania wewnętrzne programowe (wyjątki), przerwanie dla pracy krokowej, NMI, przerwanie wywoływane przepełnieniem segmentu koprocesora, INTR, wewnętrzne programowe typu INT.

Magistrala lokalna mikroprocesora 80286 jest używana przez jego wewnętrzne układy oraz urządzenie zewnętrzne, żądające dostępu do niej sygnałem HOLD =1. Priorytety operacji na magistrali lokalnej, poczynając od najwyższego, są następujące:

- przesłania, podczas których jest utrzymywany $\overline{LOCK}$ = 0 (rozkazy poprzedzone przedrostkiem *LOCK*, dostępy do deskryptora segmentu, cykle potwierdzania przerwania, rozkaz *XCHG*);
- drugi cykl przesłania słowa z adresem nieparzystym;
- drugi lub trzeci cykl przesłania danych koprocesora;
- żądanie dostępu do magistrali lokalnej z innego modułu nadrzędnego, generującego sygnał HOLD = 1;
- przesłanie argumentu danych do koprocesora inicjowane sygnałem PEREQ;
- przesłanie danej inicjowane przez układ wykonawczy EU w trakcie realizacji rozkazu;
- żądanie pobrania kodu z układu magistrali BIU. Układ wykonawczy EU będzie blokował pobranie kodu przed przesyłaniem danej na czas trwania dwóch taktów procesora (w celu zminimalizowania czasu oczekiwania układu wykonawczego na zakończenie pobrania kodu).

W przypadku wystąpienia wielokrotnych błędów, wynikających z przekroczenia stosowanych mechanizmów ochrony podczas wykonywania rozkazu, mikroprocesor 80286 przechodzi do stanu wstrzymania (ang. *shutdown*). Jest to stan podobny do stanu zatrzymania, osiąganego w wyniku wykonania rozkazu *HLT*.

W stanach tych sygnały $\overline{S1}$, $\overline{S0}$, COD/$\overline{INTA}$ są równe 0, a sygnał M/$\overline{IO}$ = 1, przy czym w stanie zatrzymywania (ang. *halt*) sygnał A1 = 1, a w stanie wstrzymania sygnał A1 = 0; sterownik magistrali nie wysyła sygnału ALE. W stanach tych może być obsługiwane żądanie magistrali HOLD oraz żądanie przesłania danej do koprocesora, inicjowane sygnałem PEREQ, przy czym przerwanie pojawiające się w stanie wstrzymania, w wyniku przekroczenia wielkości segmentu koprocesora, będzie blokowało dalszą obsługę żądania przesłania danej do koprocesora.

Mikroprocesor 80286 może być wyprowadzony z tych stanów, w wyniku pojawienia się żądania przerwania niemaskowalnego NMI lub maskowalnego INTR (jeżeli jest odblokowane) bądź przerwania spowodowanego przekroczeniem wielkości segmentu koprocesora.

Mikroprocesor 80286 może być stosowany w systemach wieloprocesorowych na zasadach podobnych do stosowanych w mikroprocesorach 8086/88. Dostęp do wspólnych zasobów systemowych jest realizowany za pomocą arbitra magistrali 82289 i układu arbitrażu. Struktury systemów wieloprocesorowych z mikroprocesorami 80286 są podobne do struktur z mikroprocesorami 8086/88.

Możliwości mikroprocesora 80286 można również rozszerzyć dołączając koprocesor arytmetyczny 80287 na zasadach podobnych do obowiązujących w mikroprocesorach 8086/88. Przykład rozwiązania jednostki centralnej z koprocesorem przedstawiono w p. 3.2.1.

2.4. Koprocesory arytmetyczne 8087 i 80287

Koprocesory (ang. *coprocessor*) są procesorami, które nie mogą pracować samodzielnie. W połączeniu z odpowiednimi procesorami umożliwiają rozszerzenie zbioru rejestrów, wprowadzenie nowych typów danych i rozkazów, zapewniając dodatkowo wykonywanie złożonych operacji arytmetycznych. Ich specyfika pracy polega na tym, że pracując równolegle pobierają z pamięci taką samą sekwencję rozkazów, jak procesor, lecz wykonują tylko te, które są dla nich przeznaczone. Wykonują więc równolegle i niezależnie rozkazy znajdujące się we wspólnej pamięci programu. Przykładami koprocesorów wprowadzonych przez firmę Intel są koprocesory: 8087 do współpracy z mikroprocesorami 8086/8088, 80287 – z procesorem 80286, 80387 – z procesorem 80386. Koprocesor 8087 może przejąć sterowanie magistral i samodzielnie pobierać dane z pamięci lub przesyłać je do pamięci. Komunikacja między koprocesorem a procesorem odbywa się za pośrednictwem wspólnej pamięci. Współpraca kopro-cesora 80287 z mikroprocesorem 80286 odbywa się na nieco innych zasadach. Wynika to z faktu, że koprocesor 80287 nie może m.in. samodzielnie zapisywać danych do pamięci. Prawidłowy przebieg współpracy zapewnia zastosowanie sterownika magistrali 82288.

Programista nie musi traktować koprocesora jako odrębnego układu, zauważa jednak znaczne zwiększenie możliwości procesora. Możliwe staje się wykonanie złożonych operacji arytmetycznych z dużą dokładnością przy użyciu prostych metod programowania.

W systemie z koprocesorem operacje arytmetyczne są wykonywane od kilkudziesięciu do kilkuset razy szybciej niż w systemie bez koprocesora. Korzystając z koprocesora należy stosować odpowiednie oprogramowanie. Ocenia się, że w obecnie istniejących programach wykorzystuje się nie więcej niż 20% możliwości, stwarzanych przez koprocesor.

Pierwsze produkowane koprocesory znacznie różniły się między sobą, co stwarzało kłopoty z "dostrojeniem" ich do współpracującego mikroprocesora. Obecnie tego rodzaju problemy w zasadzie nie istnieją. Koprocesory 8087, 80287 i 80387 nie są jednak w pełni zgodne, ponadto okazuje się, że np. zastosowanie koprocesora 80287 w systemie z mikroprocesorem 80286 nie powoduje tak znacznego zwiększenia szybkości wykonywania operacji, jak w przypadku koprocesora 8087 współpracującego z mikroprocesorem 8088.

2.4.1. Słowa liczbowe

Koprocesory 8087 i 80287 operują siedmioma rodzajami słów liczbowych
– w formacie stałopozycyjnym

- 2-bajtowym dwójkowym słowem całkowitym (rys. 2.63a – format F1);
- 4-bajtowym dwójkowym słowem całkowitym (rys. 2.63b – format F2) – krótkie całkowite;
- 8-bajtowym dwójkowym słowem całkowitym (rys. 2.63c – format F3 – długie całkowite;

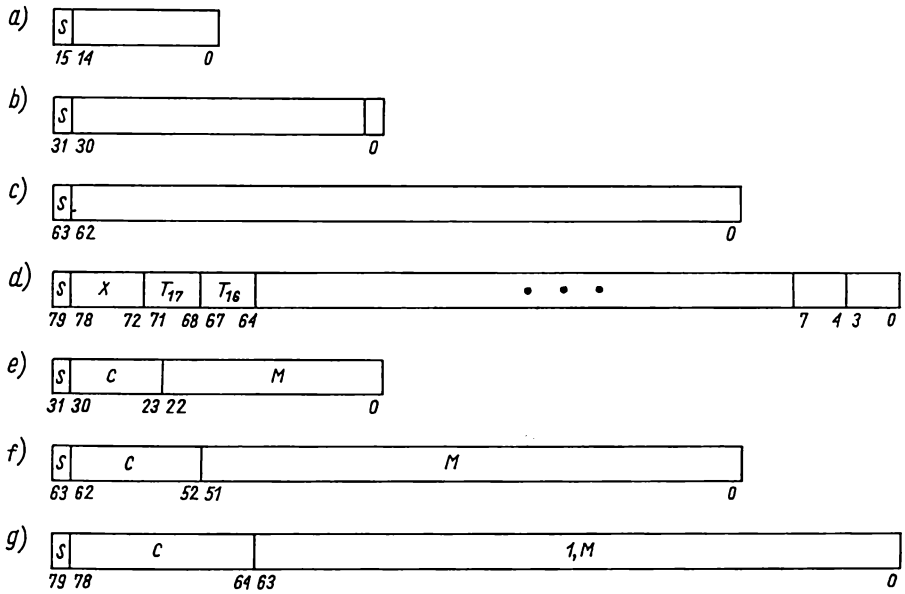
10–bajtowym upakowanym dziesiętnym słowem całkowitym BCD (rys. 2.63d – format F4) – dziesiętne upakowane;

– w formacie zmiennopozycyjnym

4–bajtowym dwójkowym słowem krótkim rzeczywistym (rys. 2.63e – format F5) – krótkie rzeczywiste;

8–bajtowym dwójkowym słowem długim rzeczywistym (rys. 2.63f – format F6) – długie rzeczywiste;

10–bajtowym dwójkowym słowem chwilowym rzeczywistym (rys. 2.63g – format F7) – chwilowe rzeczywiste.



Rys. 2.63. Formaty słów koprocesorów 8087 i 80287: S – bit znaku (0 – liczba dodatnia, 1 – liczba ujemna); X – bity nieznaczące; C – cecha (liczba całkowita); M – mantysa (ułamek); T_i – cyfra w kodzie BCD

W formacie stałopozycyjnym liczby dwójkowe

$$A = A_{n-1}...A_0 = |A| = \sum_{i=0}^{n-1} A_i \cdot 2^i$$

przedstawia się w zapisie uzupełnieniowym (znak – uzupełnienie do 2)

$$A^d = S \cdot A'_{n-1}...A'_1 A'_0 = S \cdot |A'|$$

przy czym:

$$|A'| = \begin{cases} |A| = \sum_{i=0}^{n-1} A_i 2^i & , \quad \text{gdy } S = 0 \\ |\overline{A}| = \sum_{i=0}^{n-1} \overline{A}_i 2^i + 1 & , \quad \text{gdy } S = 1 \end{cases}$$

$|\overline{A}|$ – uzupełnienie dwójkowe $|A|$,

$\overline{A}_i$ – uzupełnienie jedynkowe A_i.

Uwzględniając ten zapis można określić zakres reprezentowanych liczb dwójkowych

$$-2^n \leqslant A \leqslant 2^n - 1 \tag{2.4.1.}$$

W formacie stałopozycyjnym liczby dziesiętne są reprezentowane przez liczby 18-cyfrowe (18-tetradowe), przedstawione w zapisie modułowym (znak-moduł)

$$A^m = S \cdot T_{17} \ldots T_i\ T_{i-1} \ldots T_1\ T_0$$

przy czym:

$$T_i = t_{i3}\ t_{i2}\ t_{i1}\ t_{i0} \ldots (t_{ij} \in \{0,1\})$$

Wartość i-tej cyfry $N_i = \sum_{j=0}^{3} t_{ij} \cdot 2^j$

$$A = \sum_{i=0}^{17} N_i \cdot 10^i$$

Uwzględniając ten zapis można określić zakres reprezentowanych liczb dziesiętnych

$$-(10^{18} - 1) \leqslant A \leqslant 10^{18} - 1 \tag{2.4.2.}$$

W formacie zmiennopozycyjnym wartość liczby A otrzymuje się z zależności

$$A = (-1)^S \cdot 2^{C-P}\ \ 1,M \tag{2.4.3.}$$

przy czym:

P – przesunięcie cechy

(słowo krótkie rzeczywiste: $P = 2^7 - 1 = 127_{10} = 7F_{16}$,

słowo długie rzeczywiste: $P = 2^{10} - 1 = 1023_{10} = 3FF_{16}$,

słowo rozszerzone rzeczywiste: $P = 2^{14} - 1 = 16383_{10} = 3FFF_{16}$).

W słowach krótkim rzeczywistym i długim rzeczywistym bit części całkowitej mantysy (o wartości 1) fizycznie nie występuje (postać niejawna), występuje natomiast w słowie chwilowym rzeczywistym.

Określmy na przykład liczbę A, będącą dziesiętnym odpowiednikiem słowa krótkiego rzeczywistego, które ma postać

$$\underbrace{0}_{S}\ \underbrace{10000011}_{C}\ \underbrace{010111010\ldots0}_{M} = 41AE8000_{16}.$$

Tak więc $S = 0 \triangleq +$

$C = 10000011_2 = 131_{10}$

$P = 127_{10}$

wartość rzeczywista cechy $C - P = 4_{10}$

$1,M = 1,010111010\ldots0$

liczba $A = +2^{131-127} \cdot 1,0101111010\ldots0 = +2^4 \cdot 1,010111101 = +10101,1101_2 =$

$= +21,8125_{10}$

Z kolei spróbujmy znaleźć postać słowa krótkiego rzeczywistego, będącego odpowiednikiem liczby dziesiętnej $A = 13{,}375_{10}$.
Po przekształceniu liczby $A = 13{,}375_{10} = 1101_2 + 0{,}011_2 = 1101{,}011_2 = 1{,}101011 \cdot 2^3$
i po normalizacji $1101{,}011_2 = 1{,}101011 \cdot 2^3$
otrzymujemy: $S = 0$,
$C = 127 + 3 = 130_{10} = 10000010_2$, $M = 1010110...0_2 = 560000_{16}$.

Uwzględniając zależności (2.4.1) ... (2.4.3) można określić zakresy liczb A dla poszczególnych rodzajów słów liczbowych (tabl. 2.10).

Tablica 2.10

Rodzaj słowa liczbowego	Ilość bitów	Zakres liczb A
całkowite	16	$-32768 \ldots 32767$
krótkie całkowite	32	$-2\,147\,483\,648 \ldots 2\,147\,483\,648$ $= (\approx -2 \cdot 10^9 \ldots 2 \cdot 10^9)$
długie całkowite	64	$-9 \cdot 10^{18} \ldots 9 \cdot 10^{18}$
dziesiętne całkowite	80	$-10^{18} + 1 \ldots 10^{18} - 1$
krótkie rzeczywiste	32	$\approx \pm 8{,}4 \cdot 10^{-37} \ldots \pm 3{,}4 \cdot 10^{38}$
długie rzeczywiste	64	$\approx \pm 4{,}2 \cdot 10^{-307} \ldots \pm 1{,}7 \cdot 10^{308}$
chwilowe rzeczywiste	80	$\approx \pm 3{,}4 \cdot 10^{-4932} \ldots \pm 1{,}2 \cdot 10^{4932}$

Dzięki zastosowaniu przesuniętych cech w łatwy sposób można porównywać dwie liczby znormalizowane; porównanie kolejnych bitów od lewej do prawej kończy się w chwili wystąpienia pierwszej ich niezgodności.

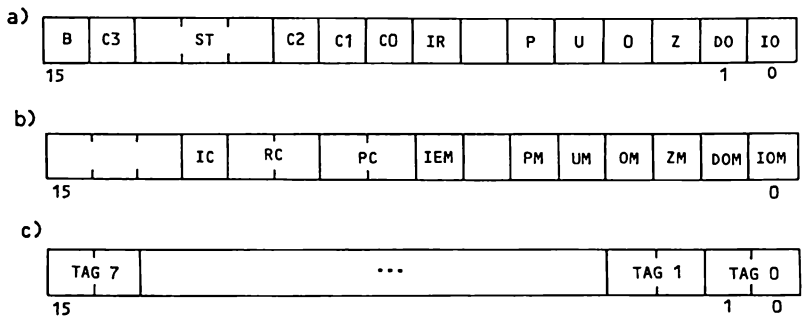
Postać chwilowa rzeczywista słowa jest używana przy wykonywaniu wszystkich wewnętrznych operacji koprocesora. Liczby, pobierane przez koprocesor z pamięci do rejestrów, są automatycznie zamieniane na postać chwilową rzeczywistą. Postacie krótka i długa rzeczywista występują jedynie w pamięci. W większości zastosowań jest niezbędne użycie postaci długiej rzeczywistej słowa (np. do zapamiętania danych i wyników). Zapewnia to wystarczający zakres i dokładność wyników przy minimalnym zaangażowaniu programisty.

Postać krótka rzeczywista słowa jest odpowiednia do zastosowań z ograniczonym rozmiarem pamięci. Przesuniętą cechę, składającą się z samych jedynek, zastrzeżono dla oznaczenia nieskończoności lub tzw. *nie liczby* (NAN – ang. *not-a-number*). Przesuniętą cechę, składającą się z samych zer, wykorzystuje się do reprezentacji zera dodatniego (gdy $S = 0$) lub zera ujemnego (gdy $S = 1$) bądź też liczby zdenormalizowanej. Liczba zdenormalizowana (ang. *denormal*) reprezentuje sobą wynik z niedomiarem, z zerami na najbardziej znaczących bitach mantysy, nawet wówczas, gdy cecha będzie skorygowana do najmniejszej z możliwych wartości. Warto jeszcze zwrócić uwagę na różnicę między liczbą zdenormalizowaną (ang. *denormal*) i nieznormalizowaną (ang. *unnormal*). W formacie zmiennopozycyjnym (postać chwilowa rzeczywista) liczby nieznormalizowane są reprezentowane przez mantysy o zakresie od 00...01 do 11...10, przy cechach o zakresie od 000...00 do 011...11.

2.4.2. Architektura

W koprocesorach 8087 i 80287 dane są przechowywane w ośmiu 80-bitowych rejestrach, tworzących stos. Zasadniczo operacje są wykonywane na zawartości wierzchołka stosu, którym może być każdy z ośmiu rejestrów.

Słowo statusu koprocesora jest przechowywane w rejestrze statusu, a słowo sterujące w rejestrze sterującym. Rejestry te są 16-bitowe.



Rys. 2.64. Formaty słów: a) stanu; b) sterującego; c) cech zawartości rejestrów danych

Znaczenie kolejnych bitów słowa statusu (rys. 2.64a)

IO (ang. *invalid operation*) – operacja niedozwolona;
DO (ang. *demormalized operand*) – argument zdenormalizowany;
Z (ang. *zerodivide*) – dzielenie przez zero;
O (ang. *overflow*) – nadmiar;
U (ang. *underflow*) – niedomiar;
P (ang. *precision*) – błąd dokładności;
IR (ang. *inerrupt request*) – żądanie przerwania;
C3...C0 (ang. *condition code*) – kod warunku;
ST2, ST1, ST0 (ang. *stack top pointer*) – wskaźnik wierzchołka stosu;
B (ang. *busy*) – zajętość (koprocesor pracuje).

Jedynki na bitach IO...P, IR i B oznaczają spełnienie danego warunku. Bity IO, DO, ..., P odgrywają niejako rolę znaczników. Bity ST2...ST0 określają, który spośród ośmiu rejestrów stosu jest aktualnie jego wierzchołkiem (ST2 ST1 ST0 = 000 – rejestr zerowy, = 001 – rejestr pierwszy, ..., = 111 – rejestr siódmy).

Po wykonaniu operacji przesłania na stos zawartość pola ST jest zmniejszana o 1, a po pobraniu ze stosu zwiększana o 1. Dla argumentu: ST oznacza wierzchołek stosu, a ST(i) – (i-ty) rejestr poniżej wierzchołka stosu, tzn. rejestr o numerze (ST) + i.

Koprocesor rozpoznaje 6 rodzajów błędów, przy czym każdy z nich można maskować (co nie spowoduje wygenerowania żądania przerwania), ustawiając w stan 1 odpowiedni bit maski w rejestrze sterującym (rys. 2.64b). Znaczenie bitów maski PM, ..., IOM jest identyczne z nazwami bitów P, ..., IO rejestru statusu, a ich oznaczenie różni się jedynie dodatkową literą M.

Możliwość generowania przerwań, włączając w to również przerwania związane ze wspomnianymi rodzajami błędów, zależy także od stanu bitu blokady przerwania IEM (ang. *interrupt enable mask*)

IEM = 1 – blokada przerwania z wyłączeniem przypadku, gdy procesor wykonuje rozkaz *WAIT;*

IEM = 0 – wystąpienie niezamaskowanego błędu powoduje generowanie przerwania z przejściem do procedury jego obsługi.

Pozostałe bity słowa sterującego mają następujące znaczenie:

PC1, PC0 – (ang. *precision control*) – kontrola dokładności (00 – 24 bity, 01 – zarezerwowany, 10 – 53 bity, 11 – 64 bity);

RC1, RC0 – (ang. *rounding control*) – kontrola zaokrąglenia (00 – do najbliższej, 01 – w kierunku $-\infty$, 10 – w kierunku $+\infty$, 11 –obcięcie);

IC – (ang. *infinity control*) – kontrola nieskończoności;

0 – oznacza, że $+\infty$ i $-\infty$ są uważane za jedną bezznakową nieskończoność (ang. *projective closure*);

1 – oznacza, że $+\infty$ i $-\infty$ są uważane za dwie nieskończoności z określonym znakiem (ang. *affine closure*).

Operacje niedozwolone, argument zdenormalizowany i dzielenie przez zero są sygnalizowane przed rozpoczęciem działania, a nadmiar, niedomiar i błąd dokładności są sygnalizowane dopiero po uzyskaniu wyniku. W pierwszym przypadku rozkaz nie jest wykonywany i nie następuje zapis do pamięci i stosu, w drugim – zapis może nastąpić.

Jeśli błędy wystąpią równocześnie, to są one sygnalizowane w następującej kolejności:

– argument zdenormalizowany (niemaskowany);
– operacja niedozwolona;
– dzielenie przez zero;
– argument zdenormalizowany (maskowany);
– nadmiar i niedomiar;
– błąd dokładności.

Koprocesor sygnalizuje błąd ustawieniem określonego znacznika w słowie statusu w stan 1, a następnie sprawdza ustawienie maski w rejestrze sterującym. Wystąpienie niezamaskowanego błędu powoduje wygenerowanie i przejście do procedury jego obsługi.

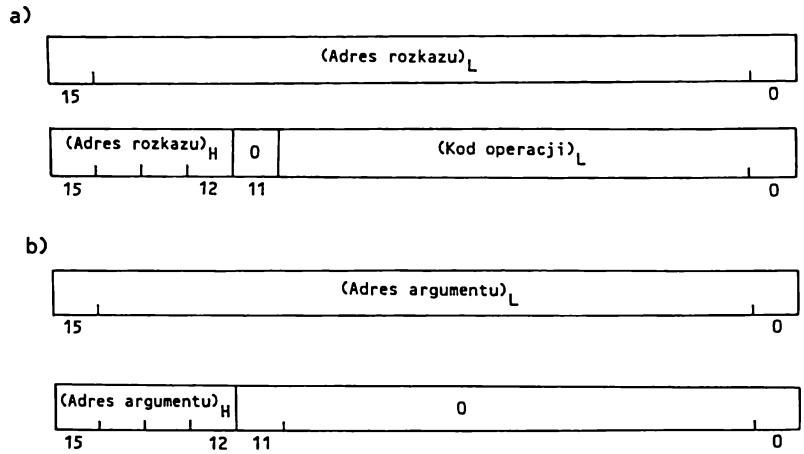
Słowo cech TAG (ang. *tag word*) zawartości rejestrów danych (rys. 2.64c) charakteryzuje zawartość tych rejestrów (TAG = 00 – liczba znormalizowana lub nieznormalizowana; 01 – zero; 10 – liczba specjalna NAN, ∞, zdenormalizowana; 11 –pusty).

Na rysunku 2.65 pokazano formaty wskaźników rozkazu i argumentu.

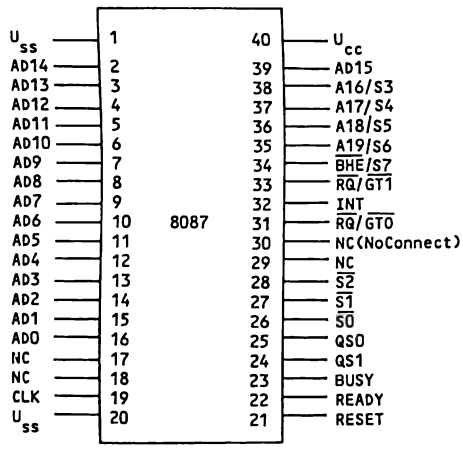
Rozmieszczenie wyprowadzeń koprocesora 8087 przedstawiono na rys. 2.66. Połączenie koprocesora 8087 z procesorem 8086 jest proste: łączy się wyprowadzenia o tych samych nazwach, a ponadto BUSY z $\overline{\text{TEST}}$, $\overline{\text{RQ}}/\text{GT0}$ z $\overline{\text{RQ}}/\text{GT1}$.

Rozmieszczenie wyprowadzeń koprocesora 80287 przedstawiono na rys. 2.67. Przy współpracy koprocesora 80287 z procesorem 80286 jest wymagane zastosowanie sterownika magistrali 82288. Łączy się ich wyprowadzenia o tych samych nazwach oraz

dodatkowo: $\overline{\text{IOWC}}$ z $\overline{\text{NPWR}}$, $\overline{\text{IORC}}$ z $\overline{\text{NPRD}}$, CLK z CLK 286, CKM z U_{CC} lub U_{SS}, NPS2 z U_{CC}.



Rys. 2.65. Formaty wskaźników: a) rozkazu; b) argumentu



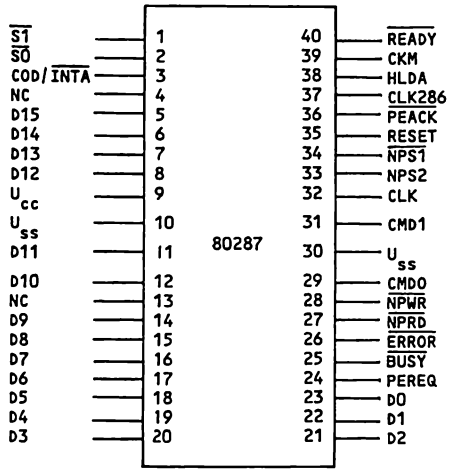
Rys. 2.66. Rozmieszczenie wyprowadzeń koprocesora 8087

Dodatkowe wyprowadzenia koprocesora 80287

CKM	– wejście trybu zegarowego;
$\overline{\text{NPRD}}$	– wejście odczytu: zezwolenie na przesłanie danych z koprocesora;
$\overline{\text{NPWR}}$	– wejście zapisu: zezwolenie na przesłanie danych do koprocesora;

$\overline{NPS1}$, NPS2 – wejścia wyboru: żadne dane nie pojawią się dopóki układ jest wybrany poprzez te wejścia;

CMD1, CMD0 – wejścia (linie) rozkazowe: wraz z wejściami wyboru umożliwiają współpracę CPU z mikroprocesorem 80287.



Rys. 2.67. Rozmieszczenie wyprowadzeń koprocesora 80287

2.4.3. Lista rozkazów

Listę rozkazów koprocesora, w zależności od realizowanych funkcji, można podzielić na sześć grup. Są to rozkazy:
- przesyłania danych;
- arytmetyczne;
- porównania;
- obliczania funkcji przestępnych;
- tworzenia stałych;
- sterujące procesorem.

Ponieważ koprocesor i procesor wybierają rozkazy występujące w programie, asembler umożliwia pisanie programów zawierających zarówno rozkazy procesora, jak i koprocesora.

W tablicy 2.11 zestawiono rozkazy koprocesora 8087, oznaczając: s – argument źródłowy, d – argument docelowy, OM – argument w pamięci, ST' – zawartość pola ST2 ST1 ST0, [a]' = entier a, [argument] – argument może wystąpić lub nie. Dla argumentu: ST oznacza rejestr wierzchołka stosu, a ST(i) – i-ty rejestr poniżej wierzchołka stosu.

Tablica 2.11

Mnemonika	Postać i format argumentów	Znaczenie	Rozpoznawanie błędów (znaczniki)
1	2	3	4
I. Grupa rozkazów przesyłania danych			
FLD s	s: ST(i) lub OM (F5...F7)	ST ← s ST' = ST' − 1	IO, DO
FILD s	s: OM (F1...F3)		IO
FBLD s	s: OM (F4)		IO
FST d	d: ST(i) lub OM (F5...F6)	d ← ST	IO, O, U, P
FIST d	d: OM (F1...F2)		IO, P
FBSTP d	d: OM (F4)	d ← ST ST' = ST' + 1	IO
FSTP d	d: ST(i) lub OM (F5...F7)		IO, DO, U, P
FISTP d	d: OM (F1...F3)		IO, P
FXCH[d]	d: ST(i), a jeśli nie występuje, to odnosi się do ST 1	d ⟷ ST	IO
II. Grupa rozkazów arytmetycznych			
FADD[[d,]s]	s, d - w postaci niejawnej; s - określony, d - niejawny; s, d - określone (F4...F6)	d ← d + s	IO, DO, O, U, P
FADDP d, s	d: ST(i), s: ST	ST(i) ← ST(i) + ST ST' = ST' + 1	IO, DO, O, U, P
FIADD s	s: OM (F1, F2)	ST ← ST + s	IO, DO, O, P
FSUB[[d,]s]	por. FADD	d ← d − s	IO, O, O, U, P
FSUBP d, s	d: ST(i), s: ST	ST(i) ← ST(i) − ST ST' = ST' + 1	IO, DO, O, U, P
FSUBR[[d,]s]	por. FADD	d ← s − d	IO, DO, O, U, P
FSUBRP d, s	d: ST(i), s: ST	ST(i) ← ST − ST(i) ST' = ST' + 1	O, DO, O, U, P
FISUB s	s: OM (F1, F2)	ST ← ST − s	IO, DO, O, P
FISUBR s	s: OM (F1, F2)	ST ← s − ST	IO, DO, O, P
FMUL[[d,]s]	por. FADD (F5, F6)	d ← d·s	IO, DO, O, U, P

Tablica 2.11 cd.

1	2	3	4
FMULP d, s	d: ST(i), s: ST	ST(i) ← ST(i) ST ST′ = ST′ + 1	IO, DO, O, U, P
FIMUL s	s: OM (F1, F2)	ST ← ST·s	IO, DO, O, P
FDIV[[d,]s]	por. FADD (F5, F6)	d ← d/s	IO, DO, Z, O, U, P
FDIVP d, s	d: ST(i), s: ST	ST(i) ← ST(i)/ST ST′ = ST′ + 1	IO, O, Z, O, U, P
FDIVR[[d,]s]	por. FADD (F5, F6)	d ← s/r	IO, DO, Z, O, U, P
FDIVRP d, s	d: ST(i), s: ST	ST(i) ← ST/ST(i) ST′ = ST′ + 1	IO, DO, Z, O, U, P
FIDIV s	s: OM (F1, F2)	ST ← ST/s	IO, DO, Z, O, U, P
FIDIVR s	s: OM (F1, F2)	ST ← s/ST	IO, DO, Z, O, U, P
FABS	—	ST ← |ST|	IO
FCHS	—	ST ← −ST	IO
FPREM	—	ST MOD ST(1)	IO, DO, U
FRNDINT	—	ST ← [ST]′	IO, P
FS CALE	—	ST ← ST·2ⁿ przy czym: n =[ST(1)]′	IO, O, P
FSQRT	—	ST ← $\sqrt{ST}$	IO, DO, P
FXTRACT	—	(Rejestr roboczy 1) ← cecha ST (Rejestr roboczy 2) ← mantysa ST ST ← (Rejestr roboczy 1) ST′ = ST′ − 1 ST ← (Rejestr roboczy 2)	IO
III. Grupa rozkazów porównania			
FCOM [s]	s: ST(i) lub OM (F5, F6); jeśli argument nieokreślony, to ST(1)	ST − s	C3, C0 - zgodnie z uwagą 1 IO, DO
FCOMP [s]	j.w.	ST − s ST′ = ST′ + 1	j.w.
FCOMPP	—	ST − ST(i) ST′ = ST′ + 2	j.w.
FICOM s	s: OM (F1, F2)	ST − s	j.w.
FICOMP s	s: OM (F1, F2)	ST − s ST′ = ST′ + 1	j.w.
FTST	—	ST − 0,0	C3, C0 - zgodnie z uwagą 2; IO, DO

Tablica 2.11 cd.

FXAM	–	–	C3, C2, C1, C0 - zgodnie z uwagą 3
IV. Grupa rozkazów obliczania funkcji przestępnych.			
FZXM1	—	$ST \leftarrow 2^{ST} - 1$ $(0 \leqslant ST \leqslant 0{,}5)$	U, P
FPATAN	—	Rejestr roboczy 1 ← ARCTG(ST(1)/ST) $0 < ST(1) < ST < \infty$ $ST' = ST' + 1$ ST ← (Rejestr roboczy 1)	U, P
FPTAN	—	ST ← Licznik tg (ST) $0 < ST < \pi/4$ $ST' = ST' - 1$ ST ← Mianownik tg (ST)	IO, P
FYL2X	—	Rejestr roboczy 1 ← $ST(1) \cdot \log_2 \lvert ST \rvert$ $0 < ST < \infty, -\infty < ST(1) < +\infty$ $ST' = ST' + 1$ ST ← (Rejestr roboczy 1)	P
FYL2XP1	—	Rejestr roboczy ← $ST(1) \cdot \log_2 \lvert ST+1 \rvert$ $0 < \lvert ST \rvert < 1 - \frac{\sqrt{2}}{2}, -\infty < \lvert ST(1) \rvert < \infty$ $ST' = ST' + 1$ ST ← (Rejestr roboczy)	P
V. Grupa rozkazów tworzenia stałych.			
FLDZ	—	ST ← 0,0	IO
FLD1	—	ST ← 1,0	IO
FLDPI	—	$ST \leftarrow \pi$	IO
FLDL2E	—	$ST \leftarrow \log_2 e$	IO
FLDL2T	–	$ST \leftarrow \log_2 10$	IO
FLDLGZ	—	$ST \leftarrow \log_{10} 2$	IO
FLDLN2	—	$ST \leftarrow \log_e 2$	IO

VI. Grupa rozkazów sterujących.

Mnemonika	Postać argumentu	Znaczenie
1	2	3
FINIT/FNINIT	—	Inicjacja koprocesora Postać słowa sterującego: 03FFH Postać słowa cech zawartości rejestrów danych: FFFFH B, P, U, O, Z, DO, IO: 0 IR: 1 ST2, ST1, ST0: 000
FDISI/FNDISI	—	IEM: 1
FENI/FNENI	—	IEM: 0
FCLEX/FNCLEX	—	B, P, V, O, Z, DO, IO: 0 IR: 1

Tablica 2.11 cd.

1	2	3
FINCSTP	–	ST' = ST' + 1
FDECSTP	–	ST' = ST' – 1
FSTSW/FNSTSW d	d: 2-bajtowy OM	d ← słowo stanu
FSTCW/FNSTCW d	d: 2-bajtowy OM	d ← słowo sterujące
FLDCW s	s: 2-bajtowy OM	rejestr słowa sterującego ← s
FSTENV/FNSTENV d	d: 14-bajtowy OM	d ← zawartość rejestru sterującego d + 2 ← zawartość rejestru stanu d + 4 ← zawartość rejestru cech zawartości rejestrów danych d + 6 ← zawartość rejestru wskaźnika rozkazu d + 10 ← zawartość rejestru argumentu
FLDENV s	s: 14-bajtowy OM	rejestr sterujący ← s rejestr stanu ← s + 2 rejestr cech zawartości rejestrów danych ← s + 4 rejestr wskaźnika rozkazu ← s + 6 rejestr argumentu ← s + 10
FSAVE/FNSANE d	d: 94-bajtowy OM	d ← zawartość rejestru sterującego zawartość rejestru stanu zawartość rejestru cech zawartości rejestrów danych zawartość rejestru wskaźnika rozkazu zawartość rejestru wskaźnika argumentu ST, ST 1, ST 2, ..., ST 7
FRSTOR s	s: 94-bajtowy OM	s ← {j.w.}
FFREE s	d: ST(i)	TAG(i) : 11
FNOP	—	ST ← ST

Uwaga 1

Relacja	C3	C0
ST > s	0	0
ST < s	0	1
ST = s	1	0
nieporównywalne	1	1

Uwaga 2

Realizacja	C3	C0
ST > 0,0	0	0
ST < 0,0	0	1
ST = 0,0	1	0
nieporównywalne	1	1

Uwaga 3

ST	C3	C2	C1	C0
+ Nieznormalizowana	0	0	0	0
+ NAN	0	0	0	1
– Nieznormalizowana	0	0	1	0
– NAN	0	0	1	1
+ Znormalizowana	0	1	0	0
+ ∞	0	1	0	1
– Znormalizowana	0	1	1	0
– ∞	0	1	1	1
+ 0	1	0	0	0
Pusty	1	0	0	1
– 0	1	0	1	0
Pusty	1	0	1	1
+ Zdenormalizowana	1	1	0	0
Pusty	1	1	0	1
– Zdenormalizowana	1	1	1	0
Pusty	1	1	1	1

3. Mikrokomputery 16–bitowe

3.1. Mikrokomputer z mikroprocesorem 8088

Najpopularniejszym mikrokomputerem z mikroprocesorem 8088 jest IBM PC/XT. Opracowany przez firmę IBM i wprowadzony na rynek w 1983 r., PC/XT jest ciągle mikrokomputerem dobrze sprzedawanym. Stanowi światowy standard w tej klasie mikrokomputerów osobistych.

Sukces swój firma zawdzięcza temu, że:

- użytkownicy mikrokomputera IBM PC/XT mogą bez trudu przejąć bogate oprogramowanie opracowane dla mikrokomputerów 8–bitowych opartych na mikroprocesorach 8080, 8085 i Z80;
- zdołano uzyskać małe koszty produkcji, a co za tym idzie i niewielką cenę sprzedaży (dzięki zastosowaniu masowo produkowanych układów scalonych z otoczenia mikroprocesorów 8080 i 8085);
- zastosowano 8–bitową magistralę danych, czyli taką jaką ma większość urządzeń zewnętrznych.

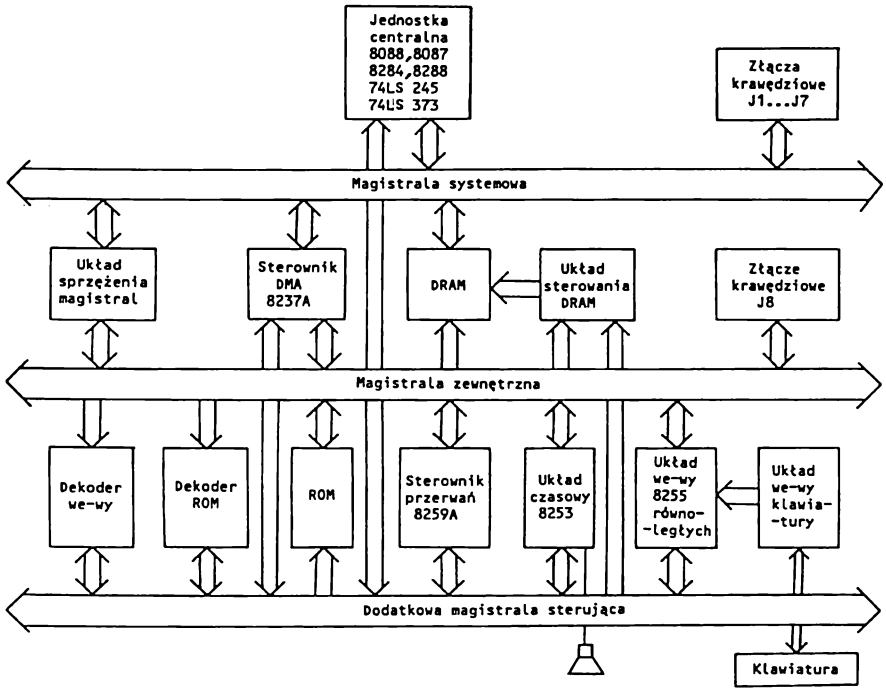
W firmach komputerowych, korzystając z tej koniunktury, rozpoczęto masową produkcję mikrokomputerów osobistych zgodnych, z punktu widzenia użytkownika, z IBM PC/XT (to znaczy takich, na które można przenieść dowolny program użytkowy z IBM PC/XT i które można rozbudować o wszystkie rozszerzenia sprzętowe opracowane dla IBM PC/XT). Wymienić tu można np. mikrokomputery M 21 firmy Olivetti i PC 10 firmy Commodore.

Cechą tych mikrokomputerów jest otwarta architektura, umożliwiająca konfigurowanie struktury systemu do określonych zadań.

3.1.1. Jednostka centralna mikrokomputera IBM PC/XT

Mikrokomputer IBM PC/XT ma strukturę modułową. Podstawowym jego *modułem* jest tzw. *płyta* (karta) *główna* (ang. *main board* lub *mother board*). Zawiera ona

jednostkę centralną (zegar 8284, mikroprocesor 8088, sterownik magistrali 8288, bufory trójstanowe, rejestry adresowe oraz opcjonalnie koprocesor 8087); sterownik przerwań 8259A; sterownik bezpośredniego dostępu do pamięci 8237A; programowany układ czasowy 8253; układ we–wy 8255 do obsługi podstawowych urządzeń zewnętrznych; pamięć RAM; pamięć ROM i do ośmiu gniazd (złączy krawędziowych – ang. *expansion slots*), umożliwiających rozszerzenie struktury o dodatkowe uniwersalne bądź specjalizowane moduły we–wy i moduły pamięci. Schemat blokowy płyty głównej przedstawiono na rys. 3.1.

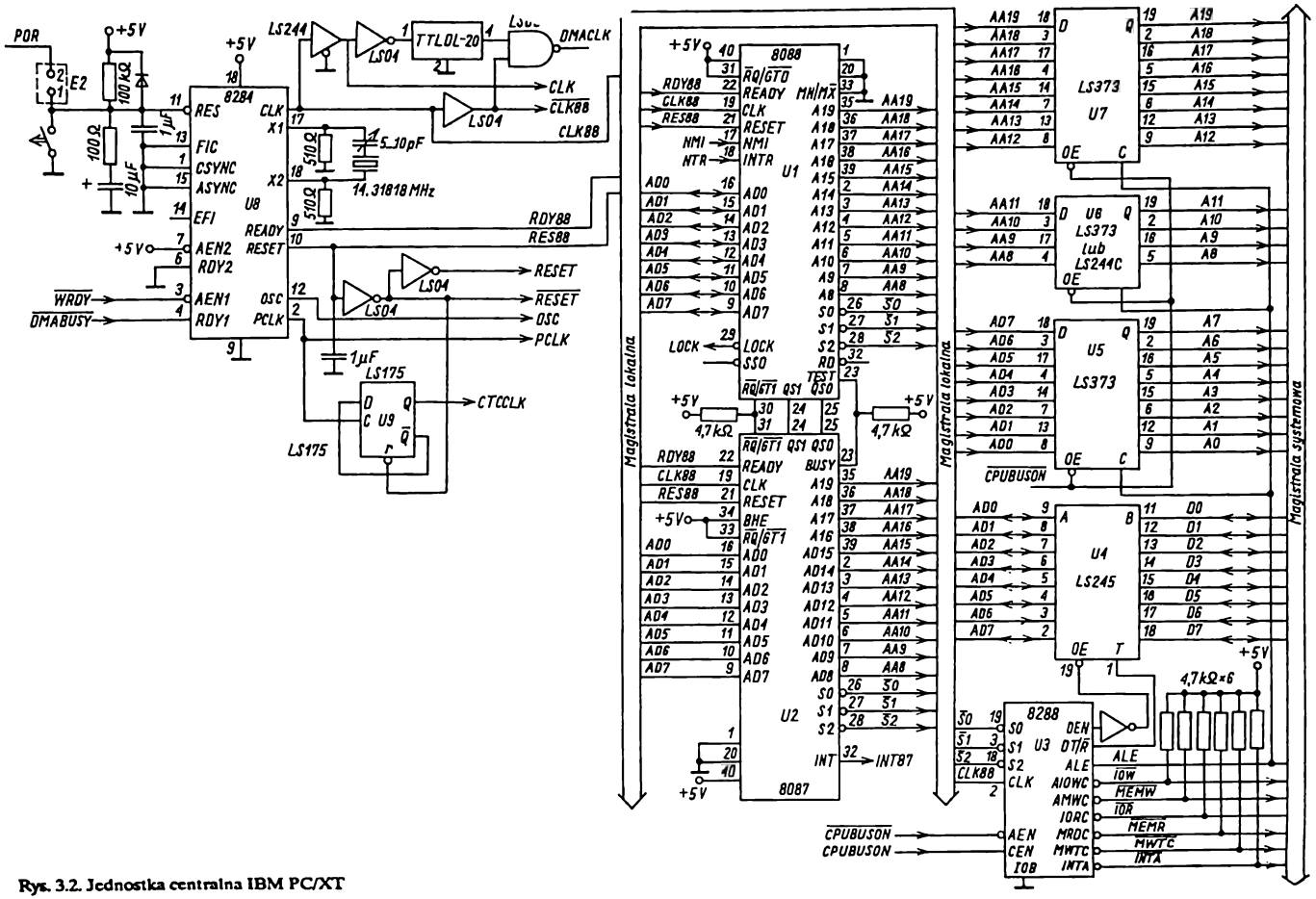


Rys. 3.1. Schemat blokowy płyty główej mikrokomputera IBM PC/XT

Układy znajdujące się na płycie głównej współpracują ze sobą za pośrednictwem czterech magistrali

- lokalnej mikroprocesora 8088 i koprocesora 8087;
- systemowej (A0...A19, D0...D7, $\overline{\text{IOW}}$, $\overline{\text{IOR}}$, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{MWTC}}$, $\overline{\text{INTA}}$);
- zewnętrznej (AX0...AX19, DX0...DX7, $\overline{\text{XIOW}}$, $\overline{\text{XIOR}}$, $\overline{\text{XMEMR}}$, $\overline{\text{XMEMW}}$);
- dodatkowej magistrali sterującej (zawierającej wszystkie pozostałe, występujące na płycie głównej, sygnały sterujące np. sygnały wyboru bloków pamięci ROM, DRAM, sygnał blokujący sterownik magistrali w czasie transmisji DMA i inne).

Jednostka centralna jest rozwiązana typowo. Jej schemat logiczny przedstawiono na rys. 3.2.



Rys. 3.2. Jednostka centralna IBM PC/XT

Mikroprocesor 8088 (U1) ustawia się w tryb pracy maksymalnej (patrz p. 2.1), dołączając wyprowadzenie MN/$\overline{MX}$ do masy. W trybie tym mikroprocesor współpracuje ze *sterownikiem magistrali* 8288 (U3), generującym podstawowe sygnały sterujące zapisem i odczytem pamięci ($\overline{MEMW}$, $\overline{MEMR}$, $\overline{MWTC}$), układów we–wy ($\overline{IOW}$ i $\overline{IOR}$), sygnał potwierdzenia przerwania INTA do *sterownika przerwań* 8259A oraz sygnały sterujące *trójstanowe bufory* 74LS245 (U4) na liniach danych i *rejestry adresowe* z wyjściem trójstanowym 74LS373 (U5, U6, U7).

W niektórych rozwiązaniach bity adresowe AA8, AA9, AA10, AA11 są buforowane tylko bramkami trójstanowymi, gdyż w mikroprocesorze 8088 bardziej znaczący bajt adresu jest utrzymywany podczas całego cyklu magistrali. Sygnał $\overline{IOWC}$ z układu 8288 jest zwykle nie wykorzystywany (w niektórych rozwiązaniach również sygnał $\overline{MWTC}$).

Sterownik magistrali ma zwarte wejście IOB do masy (brak lokalnej magistrali we–wy), co oznacza, że wszystkie odwołania do pamięci i układów we–wy odbywają się za pośrednictwem magistrali systemowej. Tylko sterownik przerwań, w niektórych rozwiązaniach płyty głównej, jest bezpośrednio dołączony do lokalnej magistrali danych. Pracuje on wówczas w trybie pracy buforowej, a dwukierunkowego sygnału $\overline{SP}$/$\overline{EN}$ używa się do blokowania dwukierunkowego bufora danych (U4) podczas komunikowania się mikroprocesora ze sterownikiem (również w czasie trwania cyklu potwierdzenia przerwania).

Do wyprowadzeń sygnałów sterujących cyklu magistrali, generowanych przez sterownik magistrali, doprowadzono napięcie +5V przez rezystory 4,7 kΩ.

Generowanie sygnałów sterujących w układzie 8288 (U3) jest blokowane podczas przesłań DMA, sygnałem $\overline{CPU\ BUSON}$.

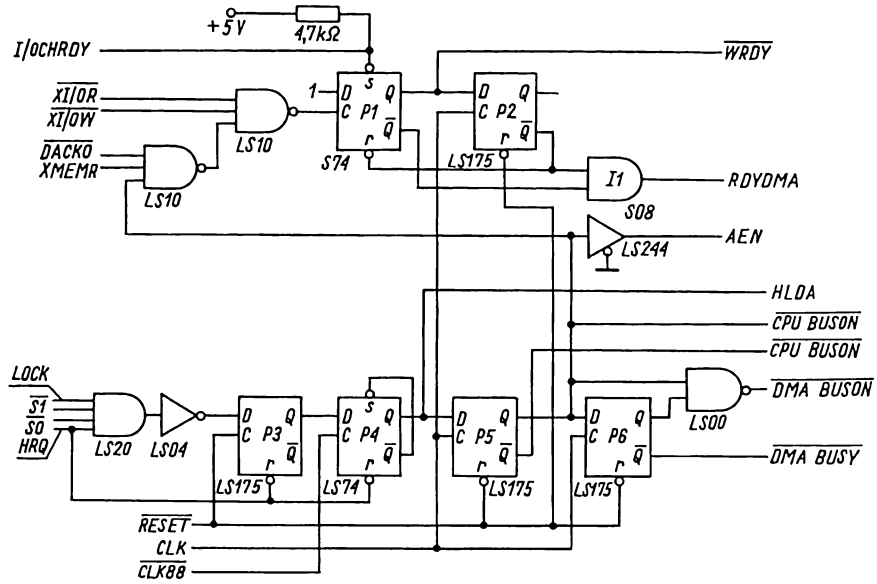
Połączenie *koprocesora arytmetycznego* 8087 (U2) z mikroprocesorem 8088 zrealizowano w klasyczny sposób – równolegle. Połączono wyprowadzenia o jednakowych oznaczeniach oraz dodatkowe wyprowadzenia: BUSY z $\overline{TEST}$ i $\overline{RQ}$/$\overline{GT0}$ z $\overline{RQ}$/$\overline{GT1}$. Do wyprowadzeń tych doprowadzono napięcie +5V przez rezystory 4,7 kΩ, w celu utrzymania ich w stanie nieaktywnym w przypadku braku koprocesora.

Praca jednostki centralnej jest synchronizowana impulsami taktującymi z układu *zegara systemowego* 8284 (U8) z rezonatorem kwarcowym o częstotliwości f = 14,31818 MHz.

Impulsy OSC o częstotliwości f doprowadzono do złączy krawędziowych dla potrzeb ewentualnych modułów dodatkowych. Pozostałe dwa rodzaje sygnałów zegarowych, CLK88 o częstotliwości 1/3 f i PCLK o częstotliwości 1/6 f, są wykorzystywane przez układy znajdujące się na płycie głównej (CLK również przez układy rozszerzające, dołączone do złączy krawędziowych).

Sygnał CLK z trójstanowego układu LS 244 zastosowano w układzie *generującym dodatkowe sygnały sterujące systemu* (rys. 3.3) oraz doprowadzono do złączy krawędziowych. Jest on również wykorzystywany do synchronizowania pracy *sterownika* DMA (8237A) jako sygnał DMA CLK. Impulsy zegarowe CLK88 synchronizują pracę mikroprocesora, sterownika magistrali i koprocesora. Ich negację również doprowadzono do układu generującego dodatkowe sygnały sterujące. Impulsy zegarowe PCLK są wykorzystywane w układzie we–wy kla-

wiatury (patrz rys. 3.14) oraz po podzieleniu przez 2 (U9) jako impulsy CTCCLK są zliczane we wszystkich 3 kanałach *układu czasowego* 8253 (patrz rys. 3.11b).



Rys. 3.3. Układ generowania dodatkowych sygnałów sterujących

Układ zegara 8284 synchronizuje ponadto sygnał WRDY, otrzymywany na podstawie sygnału gotowości układów we-wy I/OCHRDY ze złączy krawędziowych (patrz rys. 3.15) oraz sygnał zerowania systemowego.

Synchronizowany sygnał gotowości RDY88 umożliwia wprowadzenie mikroprocesora i koprocesora w stan oczekiwania przy odwołaniach do wolnych pamięci i układów we-wy oraz podczas przesłań DMA. Jest on generowany na podstawie sygnału WRDY (doprowadzonego do wejścia AEN1) i DMABUSY (doprowadzonego do RDY1). Odpowiednie wartości tych sygnałów są ustalane w układzie generowania dodatkowych sygnałów sterujących.

W cyklach zapisu lub odczytu pamięci mikroprocesor (i koprocesor) może odwoływać się do pamięci ROM i RAM, umieszczonych na płycie głównej lub do pamięci zewnętrznych, znajdujących się na modułach dodatkowych dołączonych do złączy krawędziowych (np. pamięci obrazu w modułach grafiki lub rozszerzenia pamięci podstawowej).

W odwołaniach do pamięci (RAM, ROM), znajdujących się na płycie głównej, ze względu na ich krótkie czasy dostępu, nie wymaga się wydłużenia cykli magistrali mikroprocesora. W odwołaniach do pamięci zewnętrznych może dojść do wydłużenia cykli zapisu i odczytu, gdy sygnał I/OCHRDY będzie ustawiony na zero; spowoduje to wpisanie jedynki do przerzutnika P1 (typu D) i ustawienie sygnału

$\overline{WRDY}$ na 1. Sygnał ten, doprowadzony na wejście $\overline{AEN1}$ układu 8284, spowoduje ustawienie sygnału RDY88 na 0 i wprowadzenie mikroprocesora w stan oczekiwania. Jednocześnie na wyjściu $\overline{Q}$ przerzutnika P1 i wyjściu RDYDMA (iloczyn I1–74S08) pojawi się wartość 0. Sygnał RDYDMA = 0 umożliwia wydłużenie cykli DMA.

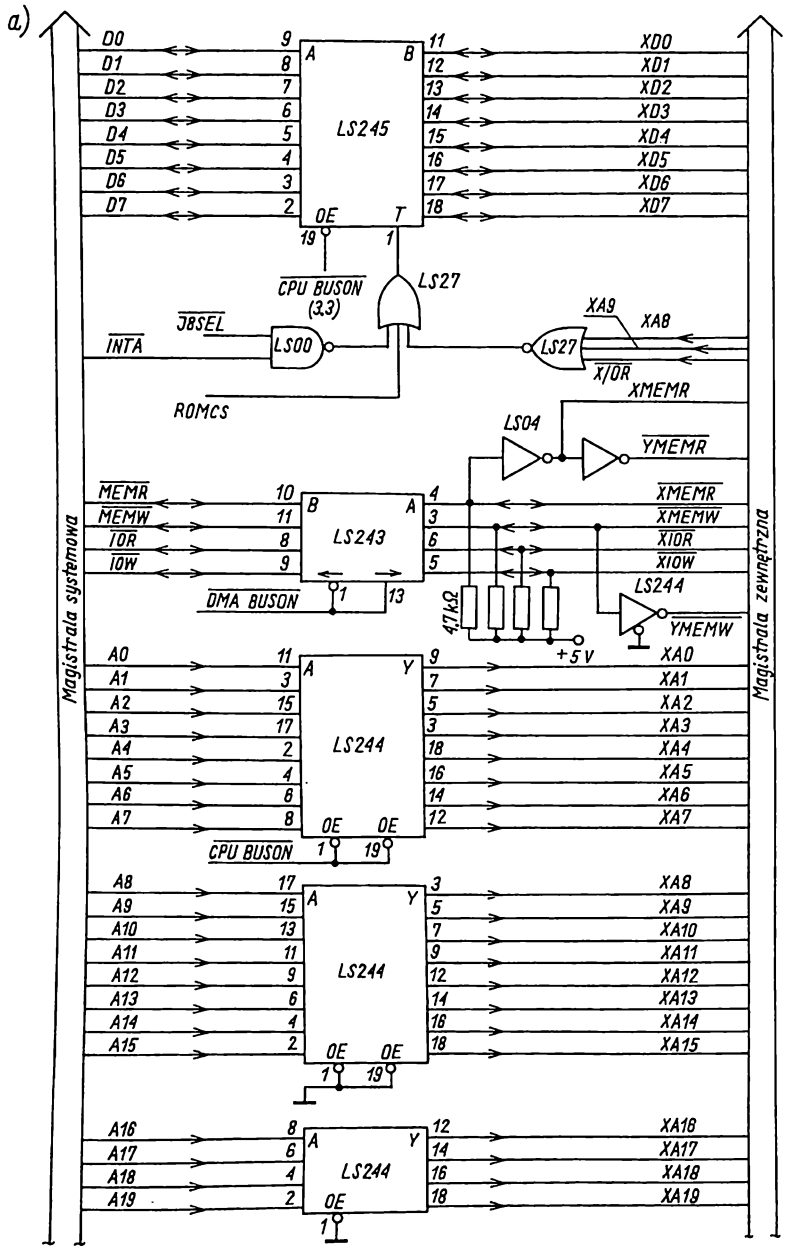
Kolejne narastające zbocze impulsu zegarowego spowoduje wpisanie jedynki (z wyjścia Q przerzutnika P1) do przerzutnika P2. Jego sygnał wyjściowy $\overline{Q}$ = 0 utrzymuje sygnał RDYDMA na poziomie niskim i jednocześnie oddziałuje na przerzutnik P1 poprzez wejście asynchroniczne $\overline{r}$. Jeżeli sygnał I/OCHRDY ma nadal wartość zero, to na obu wyjściach Q i $\overline{Q}$ przerzutnika P1 pojawią się jedynki (pracuje on jak przerzutnik asynchroniczny z dominacją wejścia wpisującego). Zapewnia to utrzymanie sygnałów $\overline{WRDY}$ = 1 i RDYDMA = 0 do chwili, gdy I/OCHRDY nie przyjmie ponownie wartości 1. Nastąpi wówczas wyzerowanie przerzutników P1 ($\overline{s}$ =1, $\overline{r}$ =0) i P2 (narastającym zboczem impulsu CLK) i uaktywnienie sygnałów gotowości $\overline{WRDY}$ = 0 do mikroprocesora (i koprocesora) oraz RDYDMA = 1 do sterownika DMA.

W cyklach we–wy mikroprocesor może odwoływać się do układów we–wy znajdujących się na płycie głównej, takich jak np. 8255 czy 8253 lub do zewnętrznych układów we–wy, takich jak np. sterownik dysków czy moduł akwizycji obrazów z kamery telewizyjnej, dołączonych przez złącza krawędziowe. Zwykle cykle te są wydłużone o 1 lub więcej taktów oczekiwania T_w. Układ generowania dodatkowych sygnałów sterujących zapewnia wydłużenie każdego cyklu we–wy o 1 takt T_w przy odwołaniach zarówno do układów we–wy, znajdujących się na płycie głównej, jak i zewnętrznych.

Pojawienie się sygnału $\overline{XIOR}$ (buforowanego sygnału $\overline{I/OR}$) w cyklu wejściowym lub sygnału $\overline{XIOW}$ (buforowanego sygnału $\overline{I/OW}$) w cyklu wyjściowym spowoduje wpisanie jedynki do przerzutnika P1. W wyniku tego nastąpi zmiana stanu sygnału $\overline{WRDY}$ na 1, co spowoduje, że sygnał RDY88 przejdzie do stanu 0, a procesor do stanu WAIT.

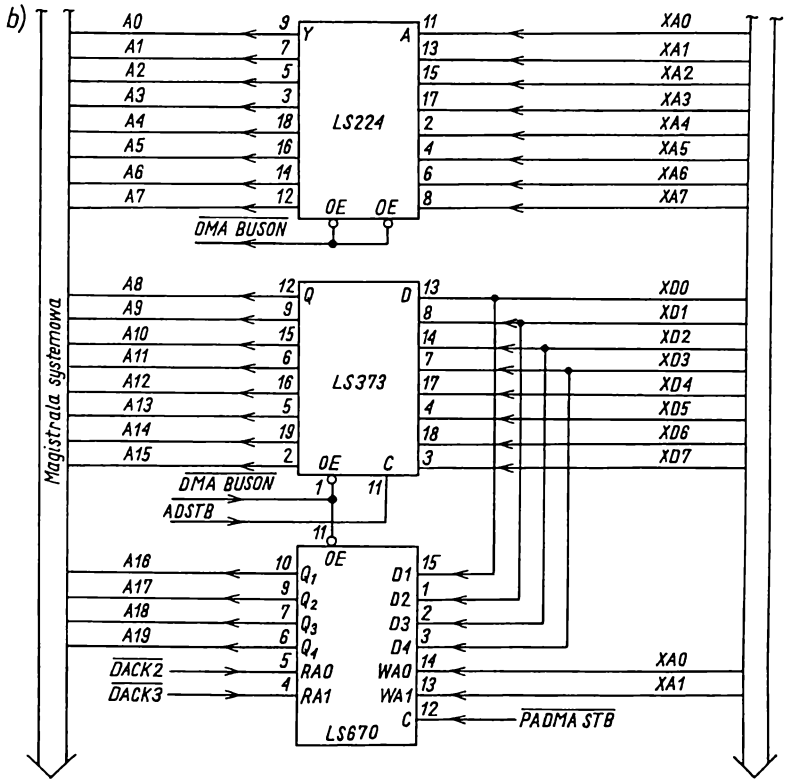
Następne narastające zbocze impulsu zegarowego CLK spowoduje przepisanie stanu 1 z wyjścia Q przerzutnika P1 do przerzutnika P2. Sygnał wyjściowy $\overline{Q}$ = 0 przerzutnika P2 spowoduje wyzerowanie przerzutnika P1 i uaktywnienie sygnału gotowości $\overline{WRDY}$ = 0 wprowadzanego na wejście $\overline{AEN1}$ generatora impulsów zegarowych i w konsekwencji – uaktywnienie sygnału gotowości mikroprocesora RDY88 = 1.

W przypadku odwołań do zewnętrznych układów we–wy, może dojść do wydłużenia cykli zapisu i odczytu, podobnie jak przy odwołaniach do pamięci zewnętrznych. Wydłużenie cykli DMA odbywa się w podobny sposób. Przy przesłaniach DMA typu pamięć → we–wy sterownik generuje sygnały $\overline{MEMR}$ i $\overline{I/OW}$, a przy przesłaniach typu we–wy → pamięć – sygnały $\overline{I/OR}$ i $\overline{MEMW}$. Zarówno cykle we–wy, jak i cykle DMA muszą być wydłużone o 1 takt oczekiwania. Zrealizowano to tak, jak w cyklach we–wy, wpisując stan 1 do przerzutnika P1 sygnałem $\overline{XIOR}$ w cyklach zapisu DMA i sygnałem XMEMR w cyklach odczytu DMA. Sygnał wyjściowy $\overline{Q}$ = 0 przerzutnika P1 powoduje ustawienie stanu sygnału RDYDMA na 0 i wydłużenie cyklu DMA o 1 takt T_w. Narastające zbocze kolejnego impulsu CLK ustawia przerzutnik P2 w stan 1, w wyniku czego następuje wyzerowanie przerzutnika P1. Kolejny impuls zegarowy zeruje przerzutnik P2, co spowoduje uaktywnienie sygnału RDYDMA = 1 i dokończenie realizowanego cyklu DMA.



Ciąg dalszy na rys. 3.4b

Rys. 3.4. Układ sprzęgania magistrali systemowej i zewnętrznej



Rys. 3.4b.

Kanał 0 sterownika DMA jest używany do realizacji cykli odświeżania (odczytu) pamięci dynamicznych DRAM (patrz rys. 3.6). Cykle te nie są wydłużane o jeden takt oczekiwania, a zapewnia to sygnał $\overline{DACK0} = 0$ wprowadzony na bramkę NAND równolegle z sygnałem XMEMR (patrz rys. 3.3). Wydłużenie cykli DMA o większą liczbę taktów może nastąpić tylko podczas przesyłania danych z zewnętrznego układu we–wy np. sterownika pamięci dyskowych do zewnętrznej pamięci np. modułu grafiki. Jest to możliwe przy ustawieniu stanu sygnału I/OCHRDY na 0. Pozostałe sygnały wyjściowe z układu generowania dodatkowych sygnałów sterujących, służą do zapewnienia prawidłowej współpracy sterownika DMA z mikroprocesorem przy przejmowaniu sterowania magistrali oraz do sterowania buforów trójstanowych, sprzęgających magistralę systemową z magistralą zewnętrzną. Sygnały te są generowane w układzie złożonym z 4 przerzutników typu D (74LS74 i 74LS175) i bramek NAND (74LS20, 74LS00, 74LS04).

Ich znaczenie jest następujące:

$\overline{\text{CPU BUSON}}$ (CPU BUSON) – sygnał aktywny poziomem niskim (wysokim)– sygnalizujący, że odbywa się cykl magistrali mikroprocesora;

$\overline{\text{DMA BUSON}}$ – sygnał aktywny poziomem niskim – sygnalizujący, że odbywa się cykl magistrali DMA.

$\overline{\text{DMA BUSY}}$ – sygnał aktywny poziomem niskim – ustawiający brak gotowości (RDY88 = 0) mikroprocesora podczas cykli DMA (patrz rys. 3.2).

Sygnał AEN jest odpowiednikiem logicznym sygnału $\overline{\text{CPU BUSON}}$ o zwiększonej obciążalności, którą zapewnia bramka trójstanowa z układu 74LS244. Doprowadzono go do złączy krawędziowych J1...J7 dla potrzeb dodatkowych modułów, dołączonych do systemu przez te złącza.

Magistrala zewnętrzna jest buforowaną wersją magistrali systemowej. Za jej pośrednictwem mikroprocesor komunikuje się z układami znajdującymi się na płycie głównej, z pamięcią ROM oraz z dowolnym modułem dodatkowym dołączonym do złącza krawędziowego J8.

Układy pośredniczące tzw. *układy sprzężenia magistrali*, występujące między magistralą systemową a zewnętrzną, przedstawiono na rys. 3.4. Są one sterowane sygnałami generowanymi w układzie przedstawionym na rys. 3.3. Linie danych magistrali zewnętrznej XD0...XD7 są sprzęgane z liniami danych magistrali systemowej D0...D7 za pomocą układu dwukierunkowych buforów trójstanowych 74LS245. Układ ten uaktywnia się sygnałem $\overline{\text{CPU BUSON}}$ = 0 podczas wszystkich cykli mikroprocesora, a blokuje sygnałem $\overline{\text{CPU BUSON}}$ = 1 podczas cykli DMA.

Kierunek przesyłania danych z magistrali systemowej do zewnętrznej lub odwrotnie zależy od realizowanego cyklu magistrali (zapisu lub odczytu). Normalnie otwarte są bufory trójstanowe, umożliwiające przesyłanie danych z magistrali systemowej do zewnętrznej podczas operacji zapisu (wejście kierunku T = 1). Dopiero w cyklach odczytu i potwierdzenia przerwania na wejściu T ustala się stan 0, co umożliwia przesyłanie danych z magistrali zewnętrznej do systemowej.

Sygnał kierunku T, generowany na wyjściu bramki NOR 74LS27, przechodzi do stanu 0 podczas cykli:

- potwierdzenia przerwania ($\overline{\text{INTA}}$ = 0) lub odczytu z modułu we–wy, znajdującego się w złączu J8 ($\overline{\text{J8 SEL}}$ = 0);
- odczytu pamięci ROM, znajdującej się na płycie głównej (ROMCS = 1);
- odczytu z dowolnego układu we–wy, znajdującego się na płycie głównej (XA8 = XA9 = $\overline{\text{XIOR}}$ = 0).

Linie adresowe magistrali systemowej A0...A7, A8...A15, A16...A19 są sprzęgane z liniami adresowymi AX0...AX7, AX8...AX15, AX16...AX19 za pomocą układów buforów trójstanowych 74LS244 podczas cykli magistrali mikroprocesora. Bufory trójstanowe na liniach adresowych A8...A19 są stale otwarte w kierunku magistrali zewnętrzej, natomiast bufor na liniach A0...A7 jest otwierany sygnałem $\overline{\text{CPUBUSON}}$ = 0 tylko podczas cykli magistrali mikroprocesora (podczas cykli DMA jest odłączany od magistrali zewnętrznej). Linie sygnałów sterujących $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$ magistrali systemowej sprzęgnięto z liniami odpowiadających im sygnałów sterujących $\overline{\text{XMEMR}}$, $\overline{\text{XMEMW}}$, $\overline{\text{XIOR}}$, $\overline{\text{XIOW}}$ przez dwukierunkowy bufor trójstanowy 74LS243.

Jest on otwierany od magistrali systemowej do zewnętrznej podczas cykli magistrali mikroprocesora (źródłem sygnałów sterujących jest sterownik magistrali)

i w kierunku odwrotnym podczas cykli DMA (źródłem sygnałów sterujących jest sterownik DMA).

Działanie układów sprzężenia magistrali zewnętrznej z magistralą systemową będzie przedstawione przy omawianiu sterownika DMA.

3.1.2. Podsystem pamięci RAM

Moduł pamięci RAM w mikrokomputerach IBM PC/XT zrealizowano z układów pamięci dynamicznych. Do jego budowy można stosować układy pamięci DRAM o organizacji 16 K x 1, 64 K x 1, 128 K x 1, 256 K x1 – zależnie od wymaganej pojemności od 16 KB do 640 KB pamięci RAM. Konfiguracje i rodzaj pamięci zwykle ustala się za pomocą odpowiednich przełączników i zwor.

Z poszczególnych układów pamięci DRAM utworzono matrycę zorganizowaną w 4 rzędy x 9 kolumn (w 4 bloki) zapewniając operowanie na słowach 8–bitowych i kontrolę parzystości na bicie dziewiątym.

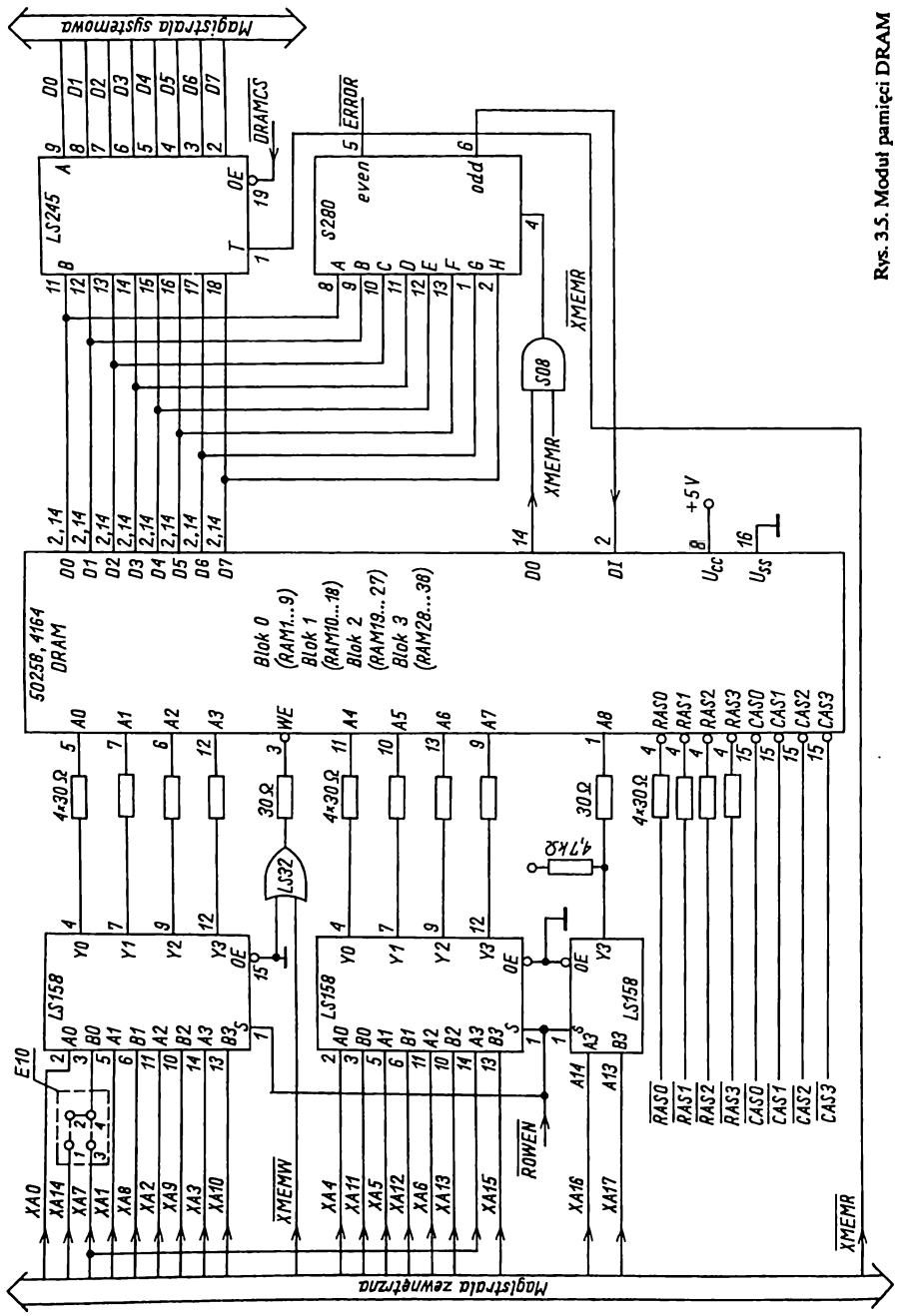
Wszystkie układy pamięci w matrycy mają 8 (9 dla układów pamięci 256 K x 1) wspólnych linii adresowych i wspólny sygnał sterujący zapisem pamięci (WE). Przykładowe rozwiązanie modułu pamięci RAM zawierającego 4 bloki przedstawiono na rys. 3.5.

Rozdzielone linie danych DIN0...DIN7 i DOUT0...DOUT7 układów pamięci zwarto (w ośmiu kolumnach) i połączono z dwukierunkowymi liniami danych D0...D7 magistrali systemowej przez bufor 74LS245, uaktywniany sygnałem $\overline{\text{DRAMCS}}$ (z układu dekodera pamięci DRAM – rys. 3.6). Kierunek przesyłania danych przez ten bufor określa się na podstawie sygnału odczytu pamięci $\overline{\text{XMEMR}}$ (z magistrali zewnętrznej).

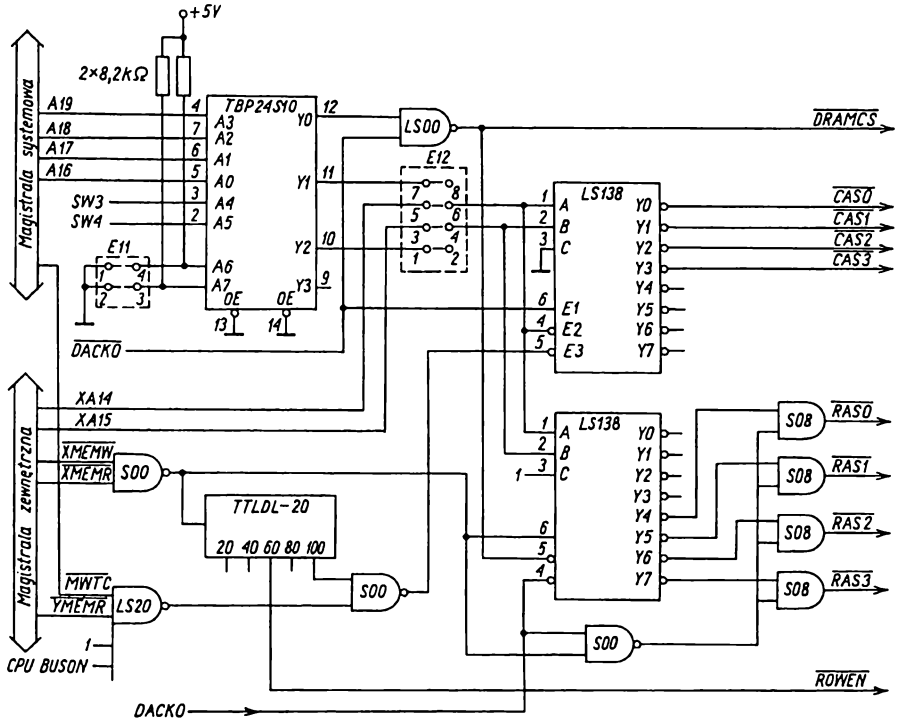
Sygnały DIN9 i DOUT9 w kolumnie 9 (oznaczone na rys. 3.5 jako DI, DO) są używane jako sygnały bitu parzystości. Przy operacjach odczytu sygnał bitu parzystości DOUT9 jest porównywany z bitem parzystości, generowanym przez układ 74S280 dla odczytywanego słowa i w przypadku niezgodności uaktywniany jest sygnał błędu ERROR i przesyłany do układu generowania przerwania niemaskowalnego (patrz rys. 3.12). Przy operacjach zapisu układ 74S280, dla każdego zapisywanego słowa, generuje sygnał bitu parzystości zapisywany do układu pamięci poprzez wejście DIN9.

Układy multiplekserów adresowych (3 x 74LS158) umożliwiają przesłanie adresu XA0...XA17 z magistrali zewnętrznej do wejść adresowych układów pamięci, z których zbudowano blok pamięci DRAM. Sygnały $\overline{\text{RAS0}}$...$\overline{\text{RAS3}}$ i $\overline{\text{CAS0}}$...$\overline{\text{CAS3}}$ sterujące adresami wierszy i kolumn w poszczególnych blokach pamięci, są generowane w układzie dekodera pamięci DRAM. Układ ten składa się z pamięci stałej PROM o organizacji 256 x 4, dwóch dekoderów 74LS138 i bramek 74S08, 74S00, 74LS20 oraz linii opóźniającej DL–20.

Dekoder generuje ponadto sygnał wyboru pamięci dynamicznej $\overline{\text{DRAMCS}}$ i sygnał adresowy $\overline{\text{ROWEN}}$ multiplekserów adresowych pamięci DRAM. Sygnał $\overline{\text{DRAMCS}}$, uaktywniający dwukierunkowy bufor trójstanowy na liniach danych, jest bramkowany sygnałem $\overline{\text{DACK0}}$ = 0 podczas cykli odświeżania pamięci. Sygnał $\overline{\text{ROWEN}}$ = 0 powoduje dołączenie linii adresowych wiersza XA0...XA7 (XA16) do



Rys. 3.5. Moduł pamięci DRAM



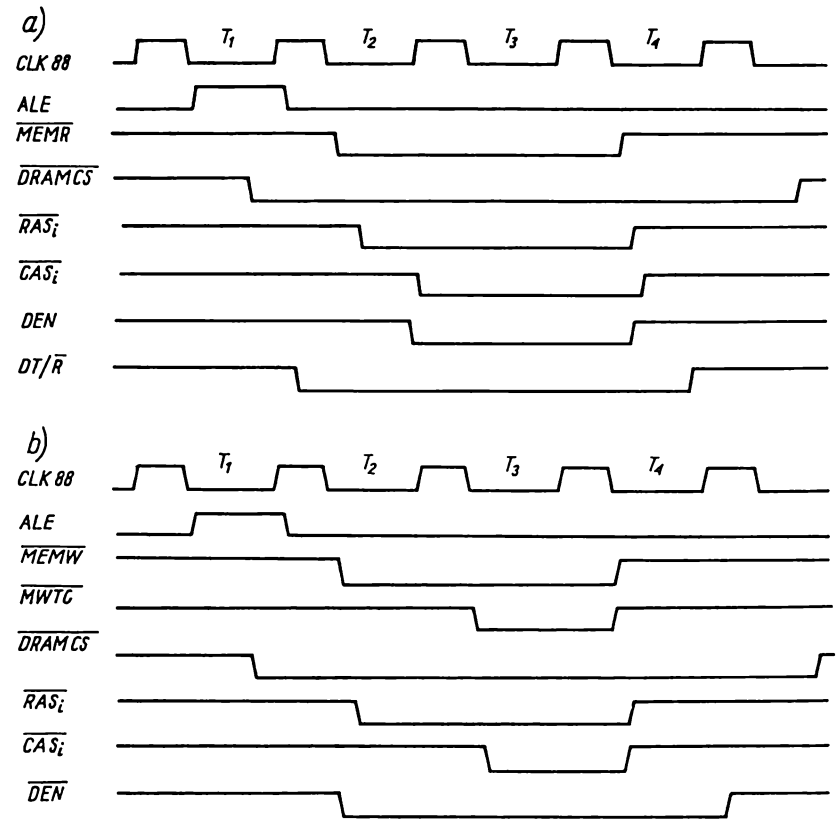
Rys. 3.6. Dekoder pamięci DRAM

modułu pamięci DRAM. Każde odwołanie do pamięci (XMEMR = 0 lub XMEMW = 0) odblokowuje dekoder sygnałów $\overline{RAS}0...\overline{RAS}3$ (przy aktywnym sygnale $\overline{DRAMCS}$ = 0 i nieaktywnym sygnale DACK0 = 0).

W zależności od ustalonego na magistrali adresowej adresu, jeden z sygnałów $\overline{RAS}_i$ (i = 0, 1, 2, 3) przechodzi do stanu 0, co spowoduje wpisanie adresu wiersza do wybranego, aktywnym sygnałem RAS_i, rzędu pamięci. Jednocześnie stan sygnału na wejściu linii opóźniającej DL–20 zmienia się z 0 na 1. Po 60 ns stan pierwszego sygnału wyjściowego linii opóźniającej zmienia się z 0 na 1 (ROWEN = 0 → 1), co powoduje zmianę sygnału adresowego multiplekserów (74LS158) i dołączenie linii adresowych kolumny XA8...XA15 (XA17) do wejść adresowych pamięci. Po upływie kolejnych 40 ns stan drugiego sygnału wyjściowego linii opóźniającej zmienia się z 0 na 1, co powoduje:

- podczas cyklu DMA (CPU BUSON = 0), uaktywnienie dekodera sygnałów $\overline{CAS}0...\overline{CAS}3$ (przy $\overline{DRAMCS}$ = 0 i $\overline{DACK0}$ = 1) i wpisanie adresu kolumny do wybranego, aktywnym sygnałem $\overline{CAS}_i$ = 0, rzędu bloku pamięci. Aktywny sygnał $\overline{RAS}_i$ zmienia stan na 1 na narastającym zboczu sygnału $\overline{XMEMR}$ ($\overline{XMEMR}$ = 0 → 1). Sygnał $\overline{CAS}_i$ = 0 jest utrzymywany dopóki sygnał $\overline{YMEMR}$ nie przejdzie do stanu 1.

– podczas cyklu zapisu pamięci przez mikroprocesor, uaktywnienie dekodera sygnałów $\overline{CAS0}$...$\overline{CAS3}$ dopiero po uaktywieniu przez sterownik magistrali sygnału $\overline{MWTC}$ = 0. Zapewnia to ustalenie danych, których źródłem jest mikroprocesor. Sygnał $\overline{RAS_i}$ = 0 zmienia stan na 1 na narastającym zboczu sygnału XMEMW (XMEMW = 0 → 1). Sygnał $\overline{CAS_i}$ = 0 jest utrzymywany, dopóki sygnał $\overline{MWTC}$ nie przejdzie do stanu 1. Sygnał $\overline{WE}$ = XMEMW przechodzi do stanu 0 w cyklach zapisu pamięci. Jest on próbkowany w bloku pamięci DRAM narastającym zboczem sygnału $\overline{CAS_i}$, w celu określenia rodzaju wykonywanej operacji (zapis dla $\overline{WE}$ = 0 bądź odczyt dla $\overline{WE}$ = 1)



Rys. 3.7. Przebiegi czasowe sygnałów podczas cyklu: a) odczytu pamięci; b) zapisu pamięci

Przykładowe przebiegi czasowe sygnałów podczas cyklu odczytu pamięci przedstawiono na rys. 3.7a, a podczas cyklu zapisu pamięci – na rys. 3.7b.

Pamięć PROM w układzie dekodera pamięci DRAM dekoduje bardziej znaczące sygnały adresowe A16...A19 (z magistrali systemowej). Poprzez dwa z jej

wyjść (zależnie od stanu sygnałów adresowych A16...A19) dekodery mogą uaktywnić jeden z sygnałów $\overline{RAS}$ lub $\overline{CAS}$, w sposób opisany wcześniej. Sygnały SW3 i SW4 z przełącznika konfiguracji (patrz rys. 3.13) służą do zadeklarowania konfiguracji i pojemności pamięci DRAM, zależnie od ustawienia zwory E11, za pomocą której określa się rodzaj zastosowanych układów pamięci.

Istnieje możliwość wyboru jednej z szesnastu możliwych konfiguracji pamięci DRAM. Przedstawiono je w tablicy 3.1.

Tablica 3.1

Stan				Pojemność pamięci	Rodzaj układu pamięci DRAM w bloku			
E11-2	E11-4	SW3	SW4	DRAM	0	1	2	3
0	0	0	0	128 KB	128 KB			
0	0	0	1	256 KB	128 KB	128 KB		
0	0	1	0	256 KB	128 KB	128 KB	128 KB	
0	0	1	1	512 KB	128 KB	128 KB	128 KB	128 KB
0	1	—	—	16 KB	16 KB			
0	1	—	—	32 KB	16 KB	16 KB		
0	1	—	—	48 KB	16 KB	16 KB	16 KB	
0	1	—	—	64 KB	16 KB	16 KB	16 KB	16 KB
1	0	0	0	256 KB	256 KB			
1	0	0	1	512 KB	256 KB	256 KB		
1	0	1	0	576 KB	256 KB	256 KB	64 KB	
1	0	1	1	640 KB	256 KB	256 KB	64 KB	64 KB
1	1	0	0	64 KB	64 KB			
1	1	0	1	128 KB	64 KB	64 KB		
1	1	1	0	192 KB	64 KB	64 KB	64 KB	
1	1	1	1	256 KB	64 KB	64 KB	64 KB	64 KB

Stan 0 zwory E11–2 (E11–4) oznacza, że połączono wejście adresowe A6 (A7) pamięci PROM z masą, a stan 1 tej zwory oznacza, że na wejście A6 (A7) doprowadzono napięcie zasilania przez rezystor 8,2 kΩ.

Przy zastosowaniu pamięci dynamicznych o pojemnościach większych bądź równych 64 K należy zewrzeć punkty 3 i 4 oraz 7 i 8 zwory E12, a przy zastosowaniu pamięci o pojemności 16 K – punkty 5 i 6 oraz 3 i 4. Dzięki temu sygnały $\overline{RAS}$ i $\overline{CAS}$ są generowane bezpośrednio na podstawie sygałów adresowych XA14 i XA15 (z pominięciem pamięci PROM). Pamięć PROM w tym przypadku służy tylko do generowania sygnału $\overline{DRMCS}$, a stan sygnałów SW3 i SW4 jest nieistotny.

Obecnie produkowane mikrokomputery IBM PC/XT zawierają zwykle pamięci DRAM zbudowane z układów pamięci o organizacji 64 K x 1 (lub 256 K x 1).

3.1.3. Podsystem pamięci ROM

Moduł pamięci ROM zawiera podstawowe procedury systemowe umożliwiające prawidłowe inicjowanie systemu, testowanie między innymi pamięci DRAM i ROM oraz obsługiwanie podstawowych układów we–wy i stacji dysków elastycznych. Niektóre mikrokomputery IBM PC/XT zawierają również dodatkowe oprogramowanie – naj-

częściej – interpretator języka Basic. W przypadku instalacji dysku stałego procedurę inicjowania systemu uzupełnia się programem rezydującym w pamięci ROM w module sterownika dysku stałego.

Po zerowaniu mikroprocesora system startuje od adresu FFFF0H. Z tego względu blok pamięci ROM umieszczono w górnej przestrzeni adresowej (1 MB) pamięci fizycznej. Blok ten może być zbudowany z pamięci ROM, EPROM lub EEPROM, umieszczanych zwykle w 5 do 8 podstawkach o 28 wyprowadzeniach.

Przykładowe rozwiązanie modułu pamięci ROM przedstawiono na rys. 3.8.

Umożliwia on również zastosowanie statycznych pamięci RAM, gdy w systemie brak pamięci dynamicznych. Moduł pamięci ROM dołączono do systemu za pośrednictwem magistrali zewnętrznej. Układ dekodera pamięci ROM, przedstawiony na rys. 3.9, zawiera dekoder 74LS138; bramki 74LS30, 74LS00 i 74LS27 oraz wzory E1, E4...E9. Zworami E4, E5 i E6 określa się, które spośród trzech sygnałów XA12...XA17 będą doprowadzone do dekodera generującego sygnały wyboru jednego z pięciu układów pamięci stałej (spotyka się również rozwiązania modułów z ośmioma układami pamięci – wówczas są wykorzystywane wszystkie wyjścia dekodera).

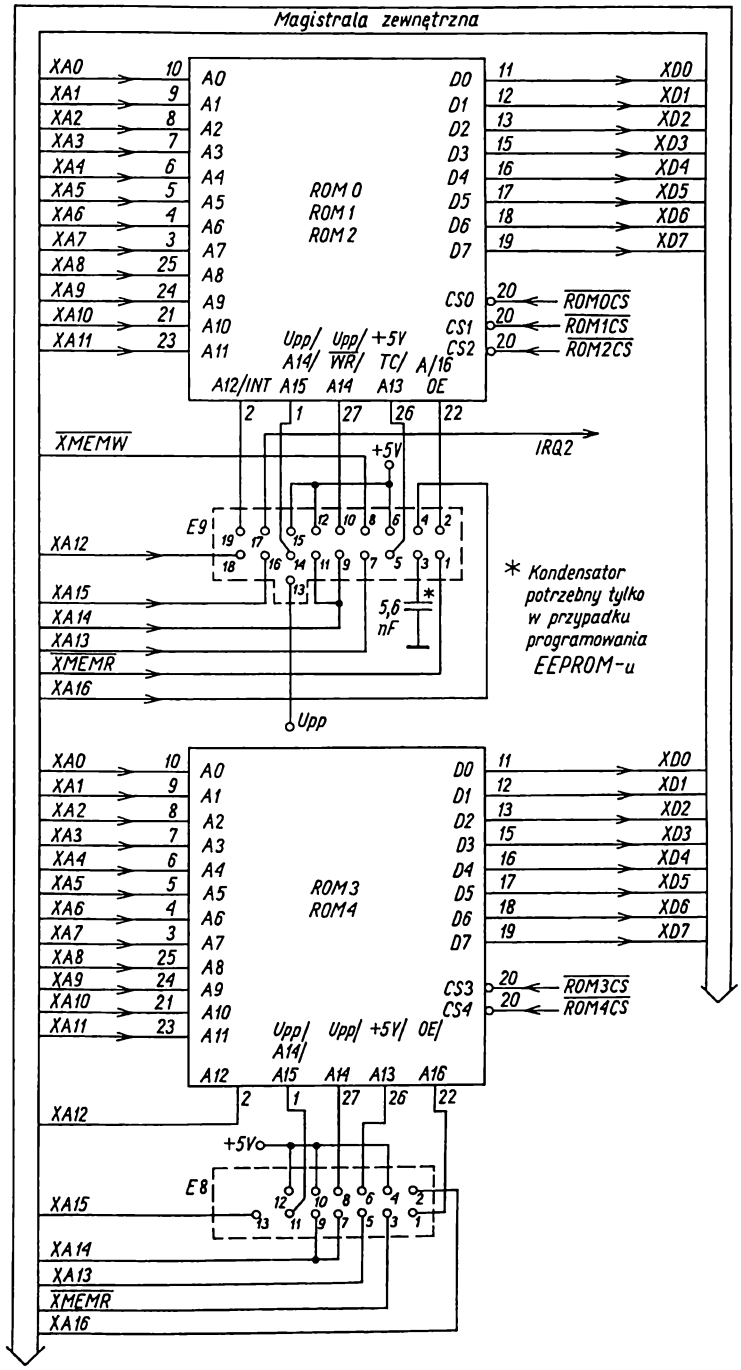
Zworę E7 stosuje się przy wyborze sygnałów wejściowych wprowadzonych na bramkę 74LS30 generującą sygnał, uaktywniający dekoder 74LS138 i sygnał ROMCS. Zwory E4, E5, E6 i E7 służą do określania obszaru pamięci ROM, przy czym zworami E4, E5 i E6 ustala się odstęp w przestrzeni adresowej między układami pamięci ROM (podstawkami) – tabl. 3.2. Zapis E_i (k–l) w tej tablicy oznacza, że zworą o numerze *i* łączy się punkt k z l. Pamięć ROM umieszczana w podstawce o numerze 4 zajmuje obszar na samym szczycie pamięci.

Tablica 3.2

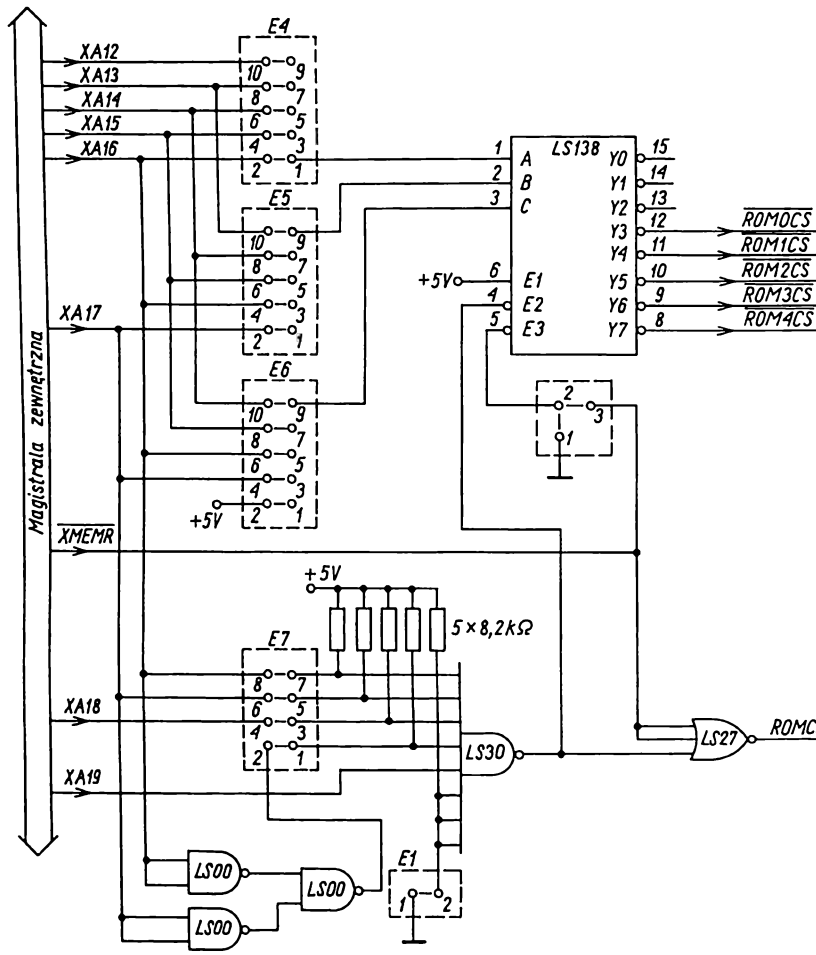
Odstęp	Pojemność pamięci ROM	Zwory
4 K	20 KB	E4 (9-10), E5 (9-10), E6 (9-10)
8 K	40 KB	E4 (7-8), E5 (7-8), E6 (7-8)
16 K	80 KB	E4 (5-6), E5 (5-6), E6 (5-6)
32 K	160 KB	E4 (3-4), E5 (3-4), E6 (3-4)
64 K	256 KB	E4 (1-2), E5 (1-2), E6 (1-2)

Przy odstępach 64 K używa się tylko pamięci umieszczonych w podstawkach o numerach od 1 do 4. Zworą E7 określa się całkowitą wielkość przestrzeni (pojemność) pamięci ROM liczoną od szczytu pamięci w dół (w kierunku malejących adresów) – tabl. 3.3. Zazwyczaj najpierw ustala się odstęp między układami pamięci, a następnie całkowitą wielkość przestrzeni pamięci ROM. Podstawki dla układów pamięci ROM o numerach 0...4 podzielono na 2 grupy: pierwsza zawiera podstawki 0, 1, 2, a druga – podstawki 3 i 4.

W podstawkach grupy pierwszej można umieszczać układy pamięci ROM, EPROM, EEPROM lub statycznych pamięci RAM – tabl. 3.4., w podstawkach grupy drugiej – układy pamięci ROM i EPROM – tabl. 3.5. Cykl odczytu pamięci ROM jest identyczny z cyklem odczytu pamięci DRAM, z tym, że nie są generowane sygnały $\overline{RAS}$ i $\overline{CAS}$.



Rys. 3.8. Moduł pamięci ROM



Rys. 3.9. Dekoder pamięci ROM

Tablica 3.3

Pojemność całkowita pamięci ROM	Zwory
64 KB	E7 (7-8), E7 (5-6), E7 (3-4)
128 KB	E7 (5-6), E7 (3-4)
192 KB	E7 (1-2)
256 KB	E7 (3-4)
512 KB	Brak zwory E7 (praktycznie nie stosowane)

Tablica 3.4

Pojemność układu pamięci stałej	Rodzaj pamięci	Zwory
2 KB	EEPROM—2817	E9 (1-2, 3-5, 8-10, 13-14, 17-19), E3 (1-3)
8 KB	RAM—6264	E9 (1-2, 5-6, 8-10, 18-19), E3 (1-3)
8 KB	EPROM—2764	E9 (1-2, 10-12, 14-15, 18-19), E3 (1-3)
16 KB	EPROM—27128	E9 (1-2, 5-7, 10-12, 14-15, 18-19), E3 (1-3)
32 KB	EPROM—27256	E9 (1-2, 5-7, 10-9, 14-15, 18-19), E3 (1-3)
8 KB	ROM—61364	E9 (1-2, 5-6, 10-12, 18-19), E3 (1-3)
16 KB	ROM—613128	E9 (1-2, 5-7, 10-12, 18-19), E3 (1-3)
32 KB	ROM—613256	E9 (1-2, 5-7, 10-9, 18-19), E3 (1-3)

Tablica 3.5

Pojemność układu pamięci stałej	Rodzaj pamięci	Zwory
8 KB	EPROM—2764	E8 (1-3, 8-10, 11-12), E3 (1-3)
16 KB	EPROM—27128	E8 (1-3, 5-6, 8-10, 11-12), E3 (1-3)
32 KB	EPROM—27256	E8 (1-3, 5-6, 7-8, 11-12), E3 (1-3)
8 KB	ROM—61364	E8 (1-3, 4-6, 8-10), E3 (1-3)
16 KB	ROM—613128	E8 (1-3, 5-6, 8-10), E3 (1-3)
32 KB	ROM—613256	E8 (1-3, 5-6, 7-8), E3 (1-3)
128 KB	ROM—62301	E8 (1-2, 5-6, 7-8, 11-13), E3 (2-3)*
8K/32 KB	ROM z 18 M PCXT	E8 (1-3, 5-6, 7-8, 9-11), E3 (2-3)

* Wymagane jest niestandardowe łączenie zwor E3-7

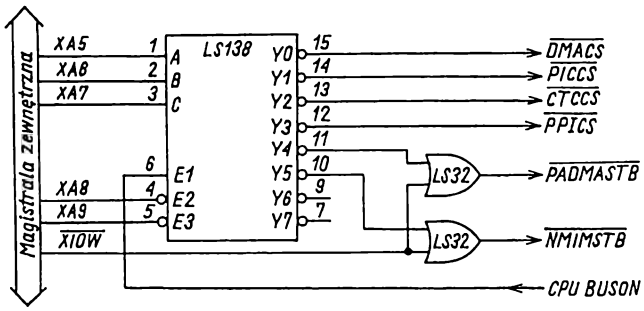
Dostępną dla procesora przestrzeń adresową (1–Mbajtową) pamięci fizycznej wykorzystano standardowo

FE000H...FFFFFH ROM z BIOS–em (ang. *basic–output system*);
F6000H...FDFFFH ROM z interpretatorem Basicu;
C8000H...C9FFFH ROM z BIOS–em dla dysku stałego (ang. *hard disk BIOS*);
C0000H...C7FFFH obszar zarezerwowany
B0000H...BFFFFH obszar pamięci obrazu;
00000H...9FFFFH pamięć RAM (640 KB)

3.1.4. Podsystem we–wy

Podstawowe układy we–wy, takie jak *sterownik przerwań*, programowany *układ czasowy*, programowany *układ we–we równoległych, rejestry stron, rejestr maski* NMI oraz *sterownik* DMA umieszczono na płycie głównej mikrokomputera. Spełniają one określone funkcje niezbędne do prawidłowej pracy mikrokomputera; przyporządkowano im stałe adresy w przestrzeni adresowej układów we–wy. Znajdujący się na płycie głównej dekoder układów we–wy (74LS138) – rys. 3.10. – zapewnia dostęp do tych układów pod następującymi adresami:

00H...0FH – sterownik DMA (8237A);
20H...21H – sterownik przerwań (8259A);
40H...43H – programowany układ czasowy (8253A);
60H...63H – programowany układ we-wy równoległych (8255A);
80H...83H – rejestry strony (74LS670);
A0H – rejestr maski przerwania NMI (74LS74).



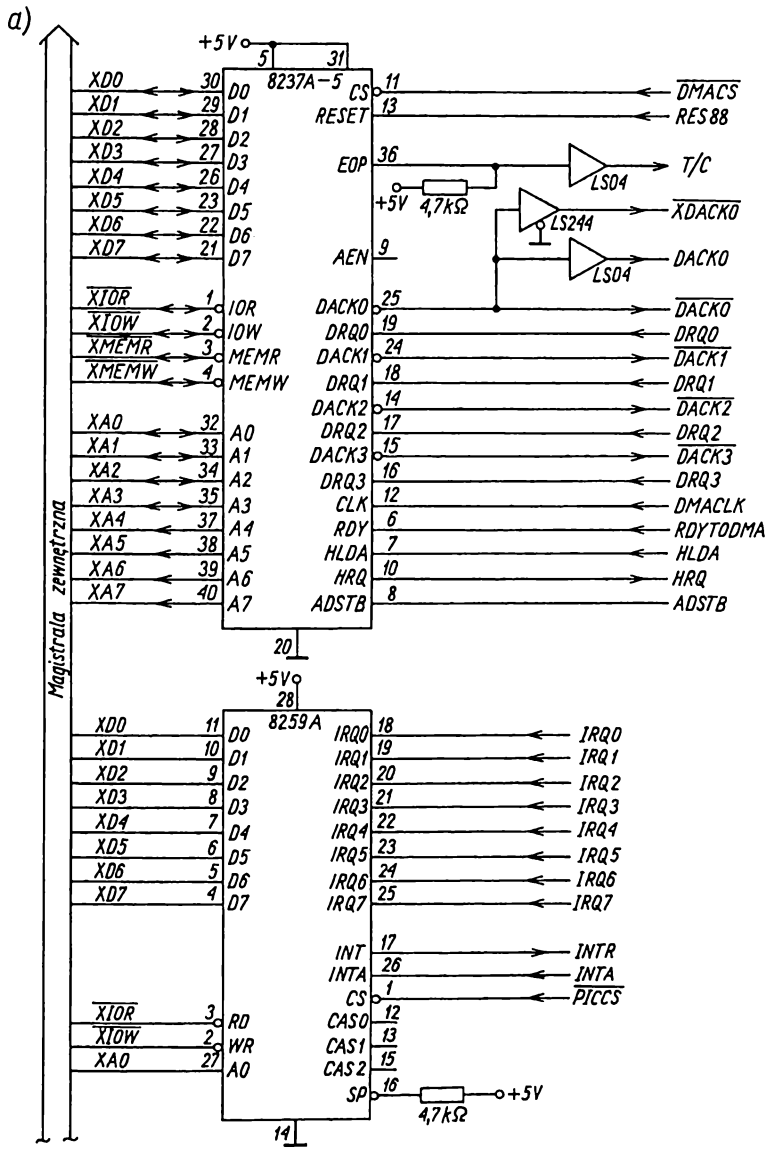
Rys. 3.10. Dekoder układów we-wy

Sterownik DMA dołączony do magistrali zewnętrznej (rys. 3.11a) zapewnia przede wszystkim
- cykliczne odświeżanie pamięci dynamicznych w kanale 0;
- szybkie przesłania danych między układami we-wy (dołączonymi przez złącza krawędziowe J1...J7) a pamięcią.

Podczas cykli DMA sterownik 8237A może wygenerować 16-bitowy adres pamięci. Mniej znaczący bajt adresu pojawia się bezpośrednio na liniach adresowych sterownika (które połączono z liniami XA0...XA7 magistrali zewnętrznej), a następnie jest przekazywany przez bufor trójstanowy 74LS244, otwierany sygnałem $\overline{\text{DMA BUSON}} = 0$ (patrz rys. 3.4b), na linie A0...A7 magistrali systemowej. Bardziej znaczący bajt adresu na liniach A8...A15 sterownik DMA wyprowadza liniami danych (dołączonymi do linii danych magistrali zewnętrznej XD0...XD7). Adres ten jest multipleksowany z danymi, musi więc być zapamiętany w rejestrze zewnętrznym 74LS373 (patrz rys. 3.4b).

Zapisanie tego adresu do rejestru zewnętrznego następuje (sygnałem ADSTB = 1) na początku każdego przesłania DMA, przy przesłaniach pojedynczych bajtów i po wystąpieniu przeniesienia z ośmiu mniej znaczących bitów adresu, przy przesłaniach blokowych. Zawartość rejestru jest przekazywana na linie A8...A15 magistrali systemowej sygnałem $\overline{\text{DMA BUSON}} = 0$, czyli podczas dostępu sterownika DMA do magistrali.

W celu zapewnienia sterownikowi DMA dostępu do całej przestrzeni adresowej pamięci w mikrokomputerze IBM PC/XT zastosowano *rejestry strony*, do których programowo, liniami danych, wpisuje się 4 najbardziej znaczące bity adresu na liniach A16...A19 – korzystając z kanałów 1, 2 i 3 (patrz rys. 3.4b). Zawartość tych bitów jest nieistotna, gdy korzysta się z kanału 0. Rejestr strony kanału 1 ma adres 83H, kanału 2 – 81H, a kanału 3 – 82H. Adres strony zapisuje się sygnałem $\overline{\text{PADMASTB}} = 0$



Ciąg dalszy na rys. 3.11b

Rys. 3.11. Układy we–wy znajdujące się na płycie głównej mikrokomputera

(z dekodera układów we–wy) do rejestru strony wybranego sygnałami adresowymi XA1 i XA0 z magistrali zewnętrznej. Wyboru rejestru strony, którego zawartość ma

być przekazana na linie adresowe A16...A19 podczas cykli DMA, dokonuje się za pomocą sygnałów $\overline{DACK2}$ i $\overline{DACK3}$.

Gdy $\overline{DACK2} = 0$ i $\overline{DACK3} = 1$, na liniach A16...A19 pojawia się zawartość rejestru strony kanału 2; gdy $\overline{DACK2} = 1$ i $\overline{DACK3} = 0$ – kanału 3, a gdy $\overline{DACK2} = \overline{DACK3} = 1$ – kanału 1. Bufory trójstanowe rejestrów strony otwiera się sygnałem $\overline{DMA\ BUSON} = 0$, podawanym na wejście $\overline{OE}$.

Przykładowy cykl przesłania danych do pamięci dyskowej, (której przydzielono kanał 2–gi) z pamięci operacyjnej, ma następujący przebieg.

W programie sterującym obsługą pamięci dyskowej programuje się sterownik, ustawia się adres początkowy, długość przesyłanego bloku i rodzaj cyklu DMA (w tym przypadku odczytu). Po zaprogramowaniu sterownik DMA czeka dopóty, dopóki układ we–wy pamięci dyskowej nie zasygnalizuje zmianą stanu sygnału DRQ2 z 0 na 1 (ang. DMA *request*) gotowości przyjęcia danych. Jeżeli nie ma zgłoszenia DRQ w kanale o wyższym priorytecie, to sterownik DMA generuje sygnał HRQ = 1 (ang. *hold request*), który doprowadzony na wejście bramki 74LS20 w układzie generowania dodatkowych sygnałów sterujących (patrz rys. 3.3) spowoduje (przy $\overline{LOCK} = 1$ i $\overline{S1}\ \overline{S0} = 11$) ustawienie przerzutnika P3 (74LS175) w stan 1, narastającym zboczem impulsu CLK. Narastające zbocze impulsu CLK88 spowoduje ustawienie przerzutnika P4 (74LS74) w stan 1. Jest on utrzymywany dopóty, dopóki stan sygnału HRQ ponownie nie zmieni się na 0. Sygnał z wyjścia przerzutnika P4 jest doprowadzany do sterownika DMA jako HLDA = 1, sygnalizując mu przejęcie sterowania magistrali.

Sterownik DMA umieszcza mniej znaczący bajt adresu na magistrali zewnętrznej, wpisuje bardziej znaczący bajt adresu do rejestru adresowego (74LS373 na rys. 3.4b) z opóźnieniem około 2 taktów w stosunku do HLDA = 1 i potwierdza urządzeniu zewnętrznemu sygnałem $\overline{DACK2} = 0$ przesłanie DMA. Na następnym narastającym zboczu CLK przerzutnik P5 (74LS175) będzie ustawiony w stan 1, co spowoduje przejście sygnału $\overline{CPU\ BUSON}$ ($\overline{CPU\ BUSON} = 1$) w stan nieaktywny, a sygnału AEN (AEN = 1) w stan aktywny. Sygnały $\overline{CPU\ BUSON} = 1$ i $\overline{CPU\ BUSON} = 0$, wprowadzone na wejścia odpowiednio $\overline{AEN}$ i CEN, zapewniają zablokowanie sterownika magistrali systemowej oraz, jak już wspomniano, zablokowanie bufora trójstanowego, umożliwiającego przekazywanie mniej znaczącego bajtu adresu z magistrali systemowej na zewnętrzną.

Sygnał AEN (o większej obciążalności niż sygnał $\overline{CPU\ BUSON}$ dzieki zastosowaniu bufora trójstanowego 74LS244) doprowadzono do wszystkich złączy krawędziowych J1...J8.

Gdy sygnał AEN = 1, wówczas informacje na magistrali adresowej podczas cyklu DMA są ignorowane przez wszystkie układy we–wy dołączone przez złącza krawędziowe. Sygnał ten służy więc do blokowania dekoderów tych układów we–wy, podobnie jak sygnał $\overline{CPU\ BUSON}$ – do blokowania dekodera układów we–wy znajdujących się na płycie głównej. Kolejne narastające zbocze impulsu CLK powoduje ustawienie przerzutnika P6 (74LS175) w stan 1, co uaktywnia sygnały $\overline{DMA\ BUSY}$ ($\overline{DMA\ BUSY} = 0$) i $\overline{DMA\ BUSON}$ ($\overline{DMA\ BUSON} = 0$). Sygnał $\overline{DMA\ BUSON} = 0$ odblokowuje rejestr adresowy bardziej znaczącego bajtu adresu, wyjścia rejestrów stron DMA, bufory trójstanowe mniej znaczącego bajtu adresu i sygnałów sterujących sterownika DMA (w kierunku od magistrali zewnętrznej do systemowej). Sygnał ten

powoduje wygenerowanie sygnału braku gotowości RDY88 = 0 w układzie zegara 8284 (patrz rys. 3.2).

W cyklu odczytu DMA sterownik DMA generuje najpierw sygnał $\overline{\text{XMEMR}}$ = 0. Dane z adresowanej komórki pamięci pojawiają się na systemowej magistrali danych. Z opóźnieniem jednego cyklu zegara pojawia się sygnał zapisu $\overline{\text{XIOW}}$ = 0 i następuje zapisanie danych do układu zewnętrznego, żądającego bezpośredniego dostępu do pamięci w kanale 2.

W cyklu zapisu DMA sterownik DMA generuje najpierw sygnał $\overline{\text{XIOR}}$ = 0. Urządzenie zewnętrzne, które otrzymało potwierdzenie przyznania kanału DMA, przesyła dane na systemową magistralę danych. Z opóźnieniem jednego cyklu zegarowego sterownik DMA generuje sygnał $\overline{\text{XMEMW}}$ = 0, powodujący przepisanie danych z magistrali systemowej do zaadresowanej komórki pamięci. Po zakończeniu przesłania, sterownik DMA steruje zmianą stanu sygnału HRQ z 1 na 0 i $\overline{\text{DACK2}}$ z 0 na 1. Sygnał HRQ = 0 zeruje przerzutnik P4 (HLDA = 0).

Narastające zbocze impulsu CLK zeruje przerzutniki P3 i P5, zapewniając odłączenie sterownika od magistrali systemowej i dołączenie mikroprocesora do tejże magistrali.

Następne narastające zbocze impulsu zegarowego ustawia przerzutnik P6 i uaktywnia sygnał gotowości RDY88 mikroprocesora, który od tej chwili może kontynuować wykonywanie programu.

Kanał 0 sterownika DMA stosuje się do realizacji cykli odświeżania pamięci dynamicznych. Korzysta się przy tym z sygnału wyjściowego OUT1 układu czasowego 8253 (rys. 3.11b). Sygnał ten osiąga stan 1 co 15,09 ms. Każde przejście stanu sygnału OUT1 z 0 na 1 powoduje ustawienie przerzutnika P7 w stan 1 i wygenerowanie sygnału żądania bezpośredniego dostępu do pamięci DRQ0 = 1. Po przyjęciu tego żądania (żądanie o najwyższym priorytecie) i przejęciu sterowania magistrali, sterownik DMA realizuje pojedynczy skrócony cykl odczytu DMA, zapewniając odświeżanie adresowanego wiersza we wszystkich blokach pamięci dynamicznej (patrz rys. 3.5 i 3.6).

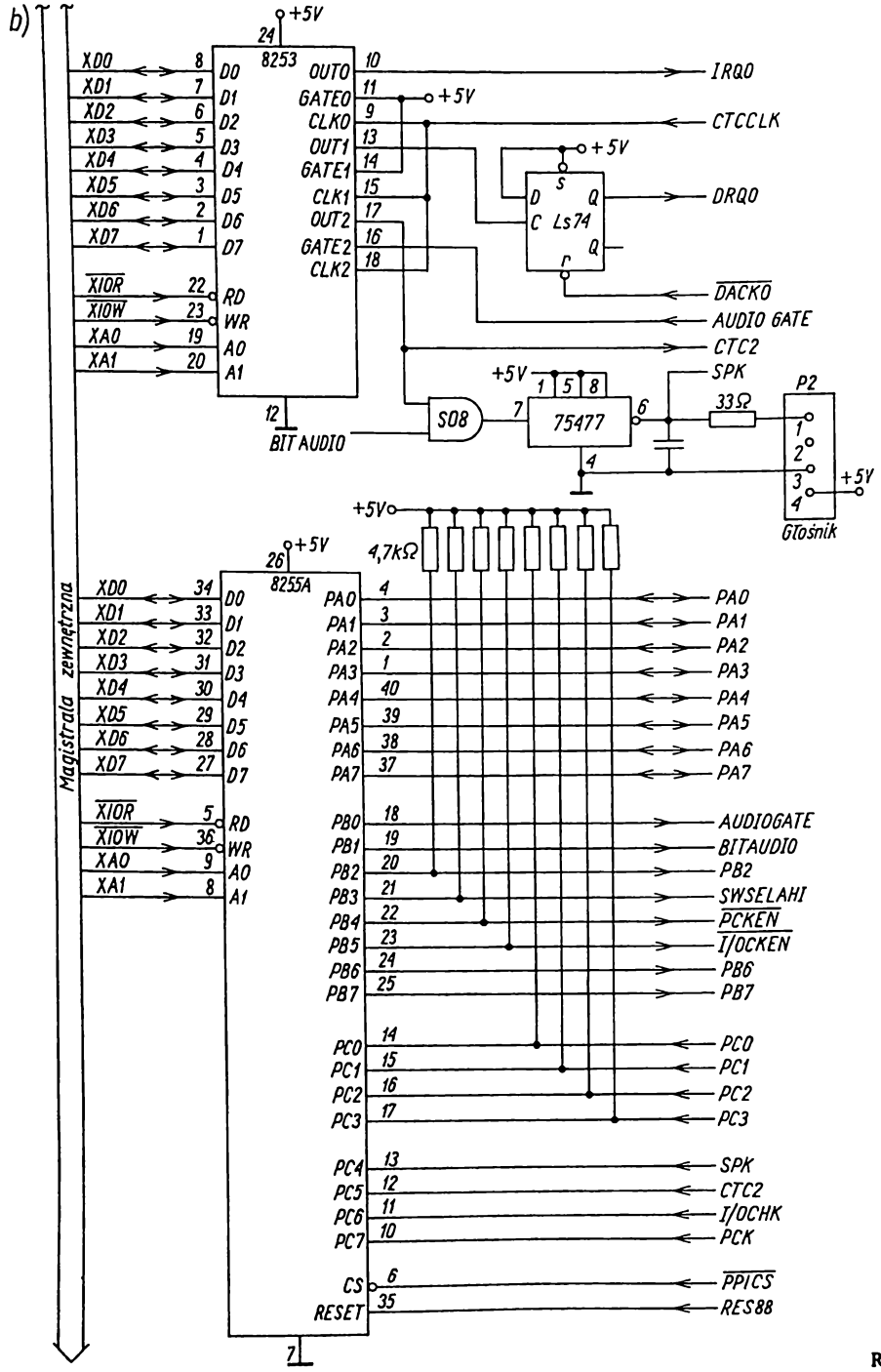
Przy iloczynie sygnałów: $\overline{\text{DACK0}}$ = 1 i $\overline{\text{XMEMR}}$ = 0 sygnał $\overline{\text{RAS0}}$ = $\overline{\text{RAS1}}$ = $\overline{\text{RAS2}}$ = $\overline{\text{RAS3}}$ = 0, co oznacza, że adres odświeżonego wiersza będzie wpisany do układów pamięci we wszystkich blokach. Negacja sygnału $\overline{\text{DACK0}}$ = 0 wprowadza bufor trójstanowy na liniach pamięci DRAM w stan dużej impedancji. Jednocześnie sygnały $\overline{\text{RAS}}$ i $\overline{\text{CAS}}$ z dekoderów 74LS138 przechodzą do stanu 1.

Mikrokomputer IBM PC/XT wyposażono w *sterownik przerwań* 8259A (rys. 3.11a) o ośmiu wejściach przerywających IRQ0...IRQ7. Wyjściowy sygnał żądania przerwania (INTR) ze sterownika 8259 wprowadza się na wejście przerwania maskowalnego mikroprocesora 8088. Wejścia przerywające sterownika mają priorytety malejące ze zwiększaniem numeru wejścia (IR0 – najwyższy, IR7 – najniższy) i są one używane do obsługi podstawowych układów we–wy (znajdujących się na płycie głównej i dołączonych przez złącza krawędziowe).

Na wejścia IRQ_i (i = 0...7) doprowadza się:

IRQ0 – sygnał żądania przerwania (OUT0) z zerowego kanału układu czasowego 8253 o częstotliwości 18,26 Hz;

IRQ1 – sygnał żądania przerwania ze sterownika klawiatury;

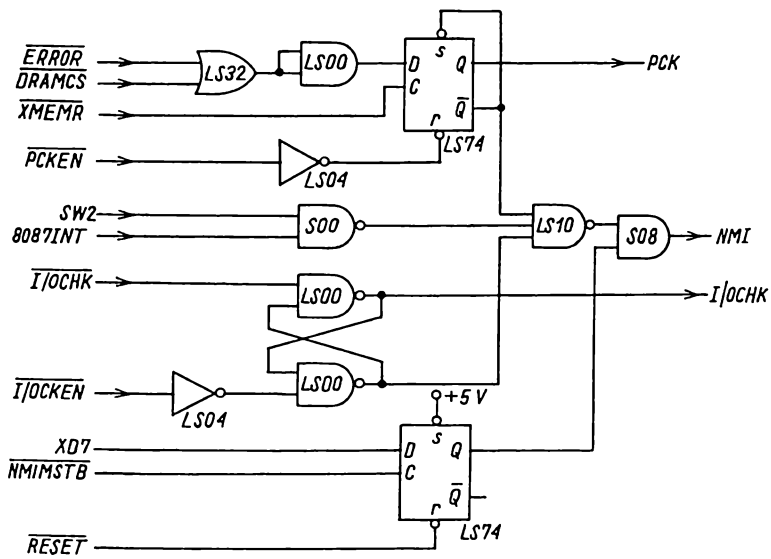

b)
Magistrala zewnętrzna
8253
+5V
XD0
XD1
XD2
XD3
XD4
XD5
XD6
XD7
XIOR
XIOW
XA0
XA1
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
OUT0
GATE0
CLK0
OUT1
GATE1
CLK1
OUT2
GATE2
CLK2
IRQ0
CTCCLK
DRQ0
Ls74
DACK0
AUDIO GATE
CTC2
SPK
S08
75477
33Ω
P2
Głośnik
BIT AUDIO
4,7kΩ
8255A
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
CS
RESET
AUDIOGATE
BITAUDIO
SWSELAHI
PCKEN
I/OCKEN
I/OCHK
PCK
PPICS
RES88

Rys. 3.11b.

IRQ2 – wejście wolne w IBM PC/XT;
IRQ3 – sygnał żądania przerwania inicjowany w programie przez COM2 (układ we–wy szeregowych);
IRQ4 – sygnał żądania przerwania (OUT0) z zerowego kanału układu czasowego 8253 o częstotliwości 18,26 Hz;
IRQ5 – sygnał żądania przerwania ze sterownika dysków stałych;
IRQ6 – sygnał żądania przerwania ze sterownika dysków elastycznych;
IRQ7 – sygnał żądania przerwania z układu we–wy drukarki.

Przerwanie niemaskowalne NMI, mające najwyższy priorytet, jest pobudzane przez układ kontroli parzystości (sygnałem ERROR = 0), układ kontroli zewnętrznych we–wy (sygnałem $\overline{I/OCHK}$ = 0) lub koprocesor arytmetyczny (sygnałem INT koprocesora 8087).

Układ generowania przerwań niemaskowalnych przedstawiono na rys. 3.12.

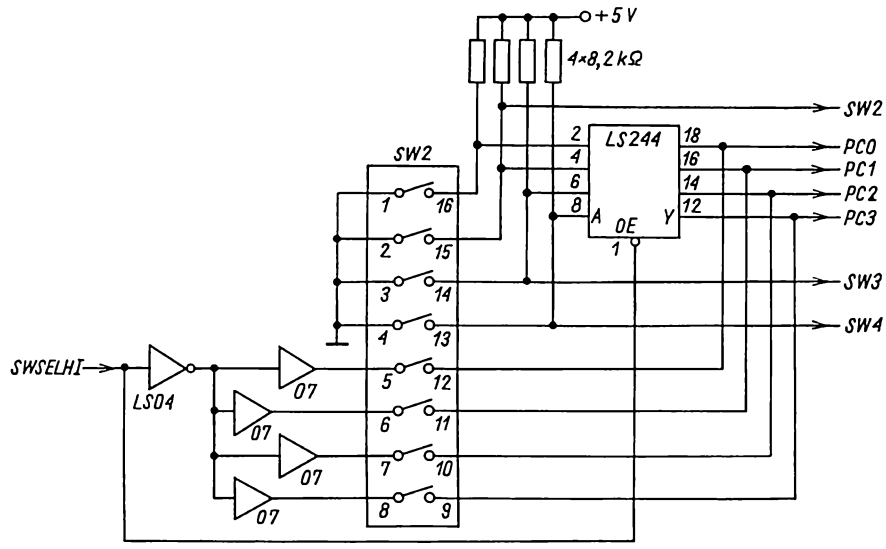


Rys. 3.12. Układ generowania przerwania niemaskowanego

Przerwanie niemaskowalne (niezależnie od przyczyny) może być zewnętrznie maskowane za pomocą jednobitowego rejestru (74LS74), dostępnego pod adresem A0H. Zmianą sygnału $\overline{NMIMSTB}$ z 0 na 1 (sygnał wyboru rejestru generowany w dekoderze układów we–wy i bramkowany sygnałem zapisu $\overline{XIOW}$ = 0) umożliwia się zapisanie do rejestru 74LS74 słowa np. 80H (XD7 = 1), którym odblokowuje się przerwanie niemaskowalne NMI. Słowem np. 00H (XD7 = 0) blokuje się przerwanie niemaskowalne. Istnieje ponadto możliwość indywidualnego blokowania każdej z trzech przyczyn generowania tego przerwania.

Przerwanie NMI, pobudzane sygnałem $\overline{ERROR}$ = 0 przy: $\overline{DRAMCS}$ = 0, $\overline{PCKEN}$ = 0 na narastającym zboczu sygnału XMEMR (podczas cyklu odczytu DRAM), można blokować sygnałem $\overline{PCKEN}$ = 1, generowanym na wyjściu PB4 układu 8255 (patrz rys. 3.11b).

Przerwanie NMI, pobudzane sygnałem koprocesora INT koprocesora 8087, można blokować sygnałem SW2 = 0, generowanym w układzie przełącznika konfiguracji systemu (rys. 3.13).



Rys. 3.13. Przełącznik konfiguracji systemu

Przerwanie NMI, pobudzane sygnałem $\overline{\text{IOCHK}}$ = 0 (rezystor zapewnia utrzymanie na wejściu przerzutnika stanu nieaktywnego 1 przy braku modułów zewnętrznych), można blokować sygnałem $\overline{\text{IOCKEN}}$ = 1, generowanym na wyjściu PB5 układu 8255. Stan sygnału IOCHK, podawany na wejście PC6 układu 8255, można testować w programie po odczytaniu portu PC.

Programowany układ czasowy 8253 zawiera 3 niezależnie programowane kanały, pobudzane sygnałami CTCCLK o częstotliwości f = 1, 93181 MHz (patrz rys. 3.11b).

Kanał 0, jak już wspomniano, jest wykorzystywany jako źródło przerwań dla zegara czasu rzeczywistego – dla potrzeb systemu operacyjnego.

Kanał 1 służy do odświeżania pamięci dynamicznych. Powoduje wygenerowanie na wyjściu przerzutnika 74LS74 żądania bezpośredniego dostępu do pamięci DRQ0 = 1 dla kanału 0 sterownika DMA (co 15,09 μs). Żądanie to można kasować sygnałem $\overline{\text{DACK}}$0 = 0.

Kanał 2 układu czasowego służy głównie do generowania sygnału dźwiękowego. Sygnał ten można blokować sygnałami AUDIO GATE = 0 lub BIT AUDIO = 0 przed podaniem go na wzmacniacz 75477, wyposażony w filtr dolnoprzepustowy. Stan sygnałów CTC2 (OUT2) i SPK można badać programowo, odczytując stan portu C układu 8255 (CTC2 = PC5, SPK = PC4) lub stosować do innych celów.

Równoległy układ wejść-wyjść 8255A (patrz rys. 3.11b) jest stosowany do obsługi sterownika klawiatury (port A i część bitów portu B), testowania stanu przełączników konfiguracji (patrz 3.13 – PC0...PC3 i PB3), stanu sygnałów błędu parzystości $\overline{\text{ERROR}}$ (PCK), CTC2, SPK i IOCHK oraz sterowania bramki NAND przenoszącej sygnał dźwiękowy (z wyjścia OUT2 układu 8253) i układu generowania przerwania niemaskowalnego NMI (sygnałami $\overline{\text{IOCKHN}}$ – patrz rys. 3.12).

Stan przełączników konfiguracji można odczytywać z wejść PC0...PC3 portu C i wyjścia PB3 portu B. Przy SWSEL = 0 bufor trójstanowy 74LS244 jest odblokowany i stan przełączników 1–16, 2–15, 3–14, 4–13 można odczytać z wejść PC0...PC3 portu C. Jednocześnie na wyjściu bramki 74LS04 pojawia się stan 1, co powoduje otwarcie odpowiednich bramek 7407, typu otwarty kolektor, przy zwarciu połączonych z nimi przełączników 5–12, 6–11, 7–10, 8–9 i ustawieniu na odpowiednich wejściach PC0...PC3 stanu nieaktywnego 1 (podobnie będzie przy rozłączeniu tych przełączników dzięki doprowadzeniu do wejść PC0...PC3 napięcia +5 V przez rezystory 4,7 kΩ).

Sygnał SWSEL = 1 blokuje bufor trójstanowy, powoduje to odłączenie przełączników od wejść PC1...PC3. Sygnał w stanie 0 na wyjściu bramki 74LS04 spowoduje zwarcie do masy każdego zamkniętego przełącznika 5–12, 6–11, 7–10, 8–9 i przekazanie tego stanu na odpowiednie wejścia PC0...PC3. Przy rozwartym przełączniku na odpowiadającym mu wejściu PC0...PC3 (podobnie jak przy SWSEL = 0) będzie utrzymywany stan 1. W ten sposób umożliwiono odczytanie stanu ośmiu przełączników przy użyciu tylko pięciu linii układu 8255.

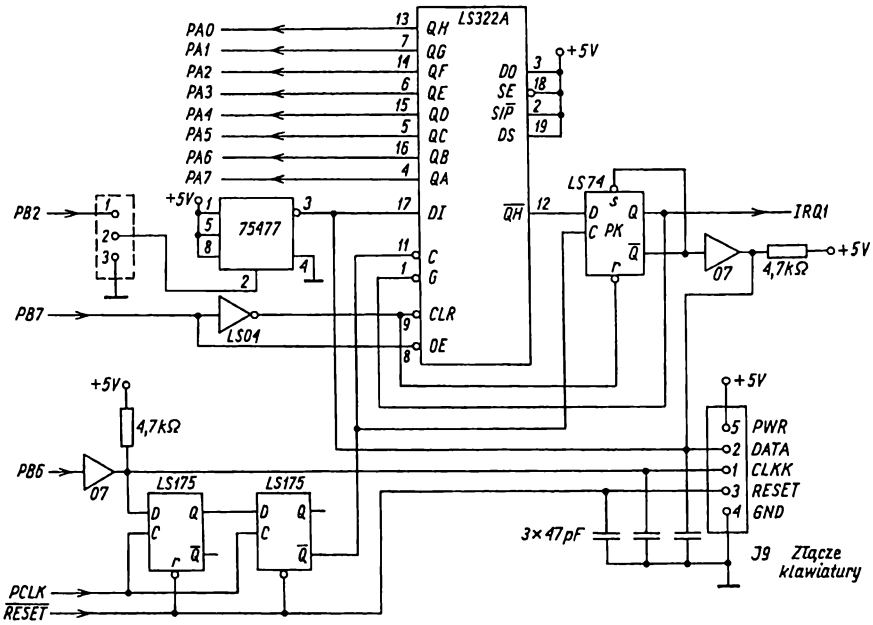
Pamięć PROM w dekoderze pamięci DRAM steruje się bezpośrednio ustawieniem przełączników 3–14 (SW3) i 4–13 (SW4) – umożliwia to zmianę jej konfiguracji.

Jak już wspomniano, podstawową funkcją układu 8255 jest pośredniczenie przy przesyłaniu danych z klawiatury. Układ we–wy klawiatury, znajdujący się na płycie głównej, służy do zamiany danych szeregowych, wprowadzanych z zewnętrznego sterownika klawiatury linią DATA ze złącza J9 (rys. 3.14), na dane równoległe.

Generuje on ponadto przerwanie IRQ1 przesyłane do sterownika przerwań 8259A po przyjęciu znaku z klawiatury. Układ we–wy klawiatury inicjuje się w programie, zmieniając stan sygnału PB7 portu B z 0 na 1 na 0. Następuje wówczas wyzerowanie rejestru przesuwnego (74LS322A) i przerzutnika żądania przerwania z klawiatury (PK) oraz ustawienie stanu 1 na linii danych DATA (tzn. uaktywnienie tej linii). Sterownik klawiatury (mikroprocesor jednoukładowy – patrz p. 3.4) inicjuje się, zerując sygnał PB6 portu B na czas 20 ms. Gdy sygnał PB6 ponownie przejdzie do stanu 1, sterownik klawiatury impulsami CLKK wpisuje do rejestru przesuwnego, kod naciśniętego przycisku, poczynając od bitu startu (transmisja asynchroniczna) i kolejno najmniej znaczących bitów kodu.

Impulsy CLKK są opóźniane w dwubitowym rejestrze przesuwnym, taktowanym impulsami zegarowymi PCLK, dzięki czemu, do rejestru przesuwnego 74LS322A są wpisywane ustalone wartości danych z linii DATA. Po ośmiu impulsach CLKK, bit startu będzie przesunięty na pozycję QH rejestru przesuwnego. Kolejny impuls CLKK powoduje usunięcie bitu startu z rejestru przesuwnego, ustawienie w stan 1 przerzutnika PK (przez przepisanie stanu 1 z wyjścia $\overline{\text{QH}}$ rejestru) i tym samym wygenerowanie żądania przerwania IRQ1, sygnalizującego procesorowi skompletowanie kodu przycis-

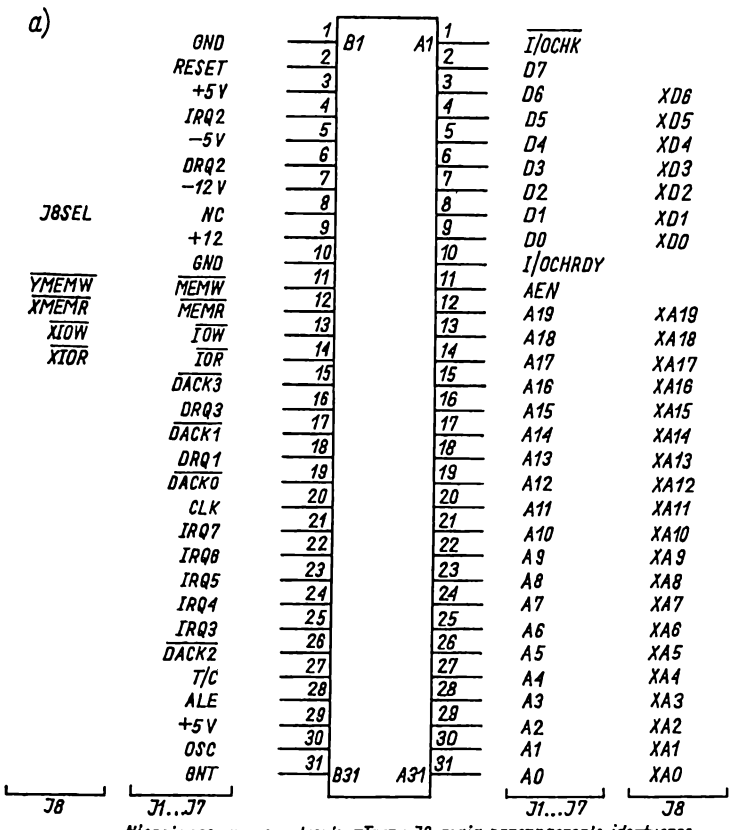
ku klawiatury (nie jest to kod znaku). Jednocześnie następuje wymuszenie stanu 0 na linii DATA przez bramkę 74LS07, sterowaną z wyjścia Q przerzutnika PK. Jeżeli sterownik klawiatury będzie nadal generował impulsy CLKK, to do rejestru będą wpisywane same zera, co traktuje się jako błąd przepełnienia. Mikroprocesor w tym przypadku będzie odczytywał same zera. W odpowiedzi na przerwanie mikroprocesor odczytuje, przez port A układu 8255, kod naciśniętego przycisku. Układ we-wy klawiatury jest ponownie inicjowany zmianą stanu sygnału PB7 z 0 na 1 na 0 i pozostaje w tym stanie, dopóki nie będzie naciścięty przycisk.



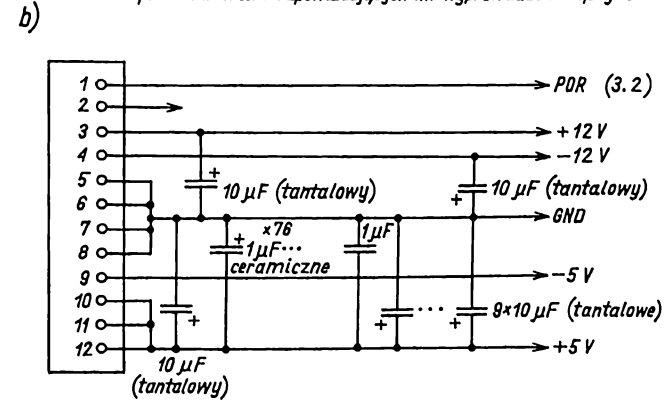
Rys. 3.14. Układ we-wy klawiatury

Gdy przycisk jest naciśnięty, sterownik klawiatury przesyła kod (pozycję) przycisku do układu we-wy klawiatury w sposób opisany poprzednio. Gdy przycisk pozostaje naciśnięty, sterownik klawiatury powtarza wysyłanie kodu przycisku w określonych przedziałach czasu. Po zwolnieniu przycisku, sterownik klawiatury ponownie wysyła kod przycisku z tym, że najbardziej znaczący bit tego kodu ma wartość 1, a nie 0 (jak przy naciśniętym przycisku). Informacja o tym, że przycisk klawiatury został zwolniony jest sygnalizowana mikroprocesorowi. Jest to istotne dla przycisków, takich jak "shift".

Jeżeli zewrze się punkty 1 i 2 zworą E13, to za pośrednictwem wyjścia PB2 układu 8255 można wymuszać, na wejściu D tego układu, określone kody przycisków (zamiast klawiatury). Taka symulacja klawiatury jest przydatna podczas testowania. Punkty 2 i 3 tej zwory zwarto fabrycznie, zapewniając prawidłową współpracę z klawiaturą, wymuszeniem stanu 1 na wyjściu układu 75477.



Nieopisane wyprowadzenia złącza J8 mają przeznaczenie identyczne z przeznaczeniem odpowiadających im wyprowadzeń złączy J1...J7



Rys. 3.15. a) Złącza krawędziowe J1...J7 i J8; b) złącze zasilacza

Na rysunku 3.15 przedstawiono opis złączy krawędziowych J1...J7 i J8 oraz złącza zasilającego P1.

W niektórych rozwiązaniach konstrukcyjnych mikrokomputerów stosuje się również dodatkowy przełącznik konfiguracji systemu SW1, za pomocą którego można deklarować rozmiar pamięci RAM (wtedy nieco inaczej rozwiązane są układy dekoderów pamięci), rodzaj monitora (kolorowy lub monochromatyczny) oraz liczbę dysków elastycznych (5,25). Stan tego przełącznika testuje się za pośrednictwem portu PA układu 8255.

Obecnie produkowane mikrokomputery IBM PC/XT zawierają znacznie mniejszą liczbę układów scalonych. Uzyskano to (przy zachowaniu pełnej zgodności funkcjonalnej z pierwowzorem) dzięki scaleniu niektórych podzespołów w jednym lub kilku układach o bardzo dużym stopniu scalenia.

3.2. Mikrokomputer z mikroprocesorem 80286

Mikrokomputer IBM PC/AT został wprowadzony na rynek w 1984 r. Jego architektura jest podobna do architektury mikrokomputera IBM PC/XT. Wykorzystyano w nim wszystkie zalety poprzednika. Zastosowano jednak znacznie nowocześniejszy mikroprocesor 80286, który zapewnia dodatkowo możliwość pracy wielozadaniowej, ochronę przydzielonych do poszczególnych zadań zasobów oraz znacznie większą szybkość wykonywania operacji. Dzięki temu oraz dzięki pełnej zgodności programowej z PC/XT, zainteresowanie tym mikrokomputerem jest bardzo duże.

Przy stosowaniu systemu operacyjnego PC-DOS olbrzymia przestrzeń adresowa mikroprocesorów 80286 nie jest wykorzystywana, gdyż system ten ogranicza pojemność pamięci dostępnej dla oprogramowania do 640 KB. Istnieje jednak możliwość rozszerzenia pojemności pamięci przez dołączenie dodatkowych bloków za pośrednictwem złączy krawędziowych.

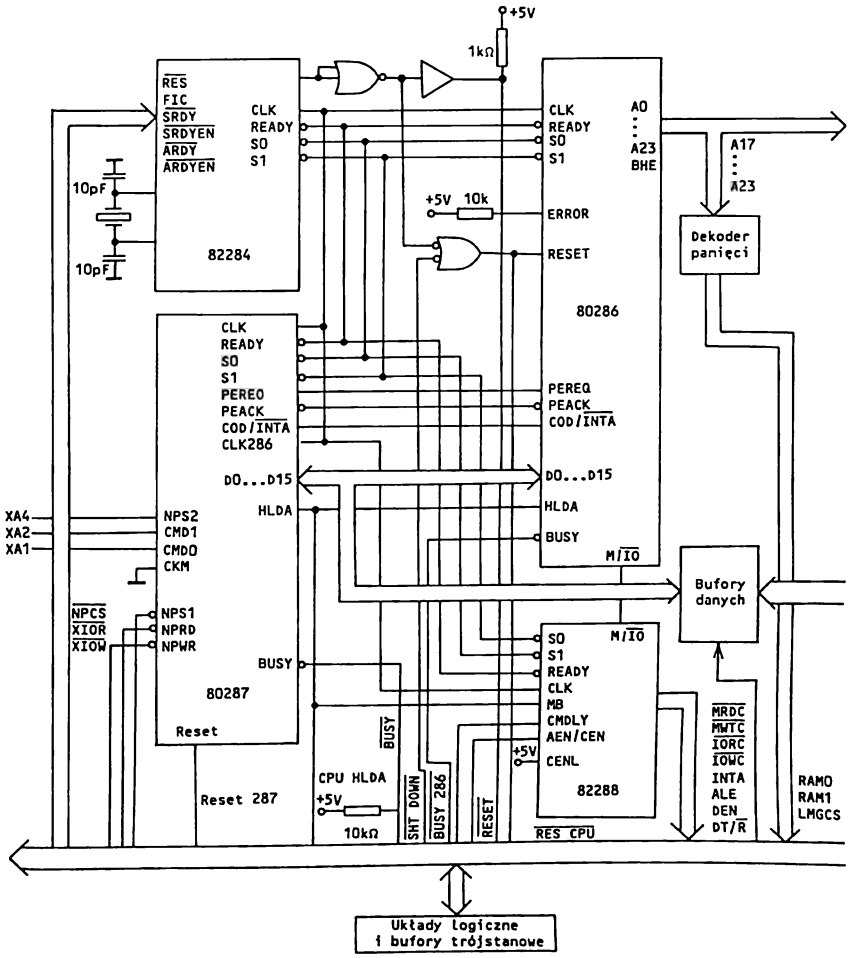
3.2.1. Jednostka centralna mikrokomputera IBM PC/AT

Mikrokomputer IBM PC/AT ma strukturę modułową. Podstawowym jego modułem (podobnie jak w IBM PC/XT) jest *płyta główna* (ang. *main board* lub *mother board*), zawierająca jednostkę centralną; 15-wejściowy sterownik przerwań (2 sterowniki 8259A w strukturze kaskadowej); 7-kanałowy sterownik bezpośredniego dostępu do pamięci (2 sterowniki 8237A w strukturze kaskadowej); programowany układ czasowy (8254A); pamięć ROM; pamięć RAM; zegar czasu rzeczywistego z pamięcią RAM (CMOS) do definiowania konfiguracji systemu; układ we-wy klawiatury; dekodery pamięci RAM, ROM i wewnętrznych układów we-wy; układ generowania przerwania niemaskowalnego; układ generowania dodatkowych sygnałów sterujących, w tym sygnału braku gotowości WAIT; 8 gniazd (złączy) krawędziowych (6 podwójnych 62- i 36-stykowych oraz 2 pojedyncze 62-stykowe.

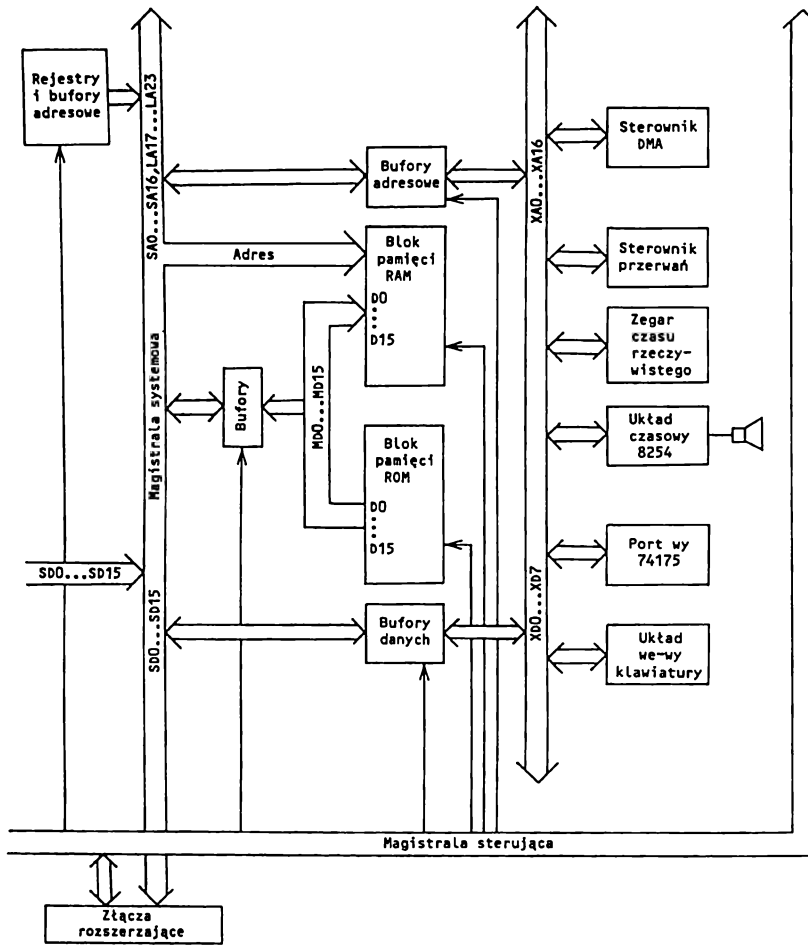
Gnizada krawędziowe, podobnie jak w PC/XT, umożliwiają rozszerzenie mikrokomputera IBM PC/AT o specjalizowane moduły we–wy i dodatkową pamięć RAM. Schemat blokowy płyty głównej przedstawiono na rys. 3.16.

W obecnie produkowanych mikrokomputerach PC/AT płyta główna zawiera zwykle kilka układów o wielkim stopniu scalenia, w których znajdują się odpowiedniki funkcjonalne układów, wyszczególnionych na rys. 3.16.

W mikrokomputerach IBM PC/AT i XT zastosowano podobne rodzaje magistral. Złącza krawędziowe, służące do rozszerzania struktury mikrokomputera, dołączono do magistrali systemowej. Wewnętrzne układy we–wy dołączono do dodatkowo buforowanej magistrali (odpowiednik magistrali zewnętrznej w PC/XT).



Rys. 3.16. Schemat blokowy mikrokomputera IBM PC/AT



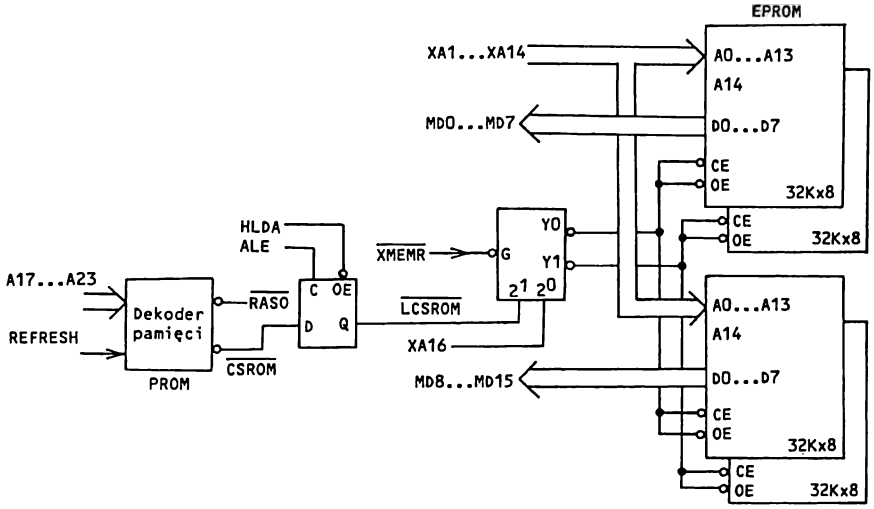
Rys. 3.16. Część II

Pamięci ROM i RAM dołączono do magistrali systemowej przez dodatkowe bufory trójstanowe. *Jednostka centralna* jest rozwiązana typowo (patrz rys. 2.54 i 3.16) i zawiera mikroprocesor 80286, układ zegara 82284, sterownik magistrali 82288, rejestry adresowe, dwukierunkowe bufory trójstanowe – na liniach danych i jednokierunkowe – na liniach sterujących oraz pomocnicze układy logiczne, generujące dodatkowe sygnały sterujące niezbędne do prawidłowej pracy mikrokomputera.

Produkowane wersje mikrokomputerów różnią się zastosowaniem innych buforów trójstanowych lub rejestrów adresowych i dopuszczalną częstotliwością pracy mikrokomputera.

3.2.2. Podsystem pamięci ROM

Pamięć ROM (rys. 3.17) o pojemności 64 KB (32 K × 16) lub 128 KB (64 K × 16) można rozszerzać o kolejne bloki (o pojemności do 128 KB) umieszczane na kartach, dołączonych przez złącza krawędziowe (kanały we–wy).

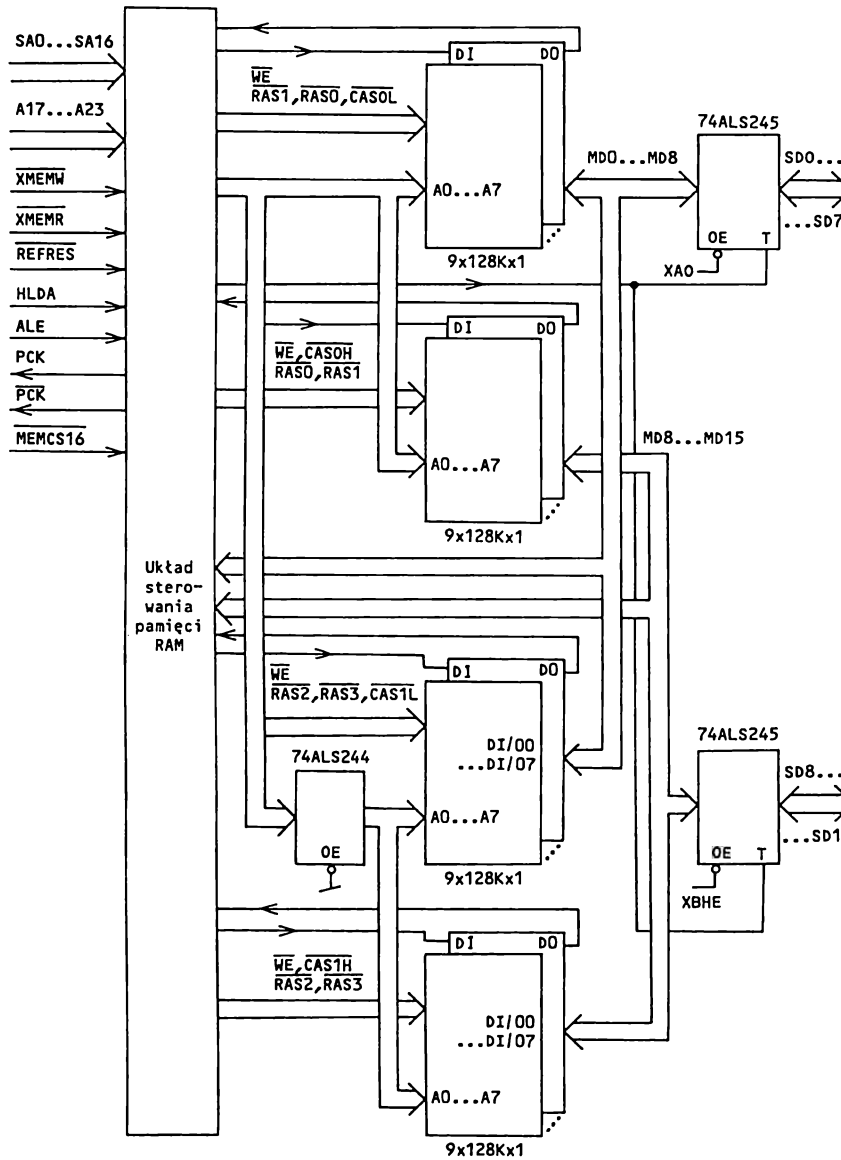


Rys. 3.17. Schemat blokowy pamięci ROM w IBM PC/AT

Do budowy pamięci ROM zwykle stosuje się układy typu EPROM 27128, 27256, 27512 o organizacji odpowiednio 16 K × 8b, 32 K × 8b, 64 K × 8b. Kody rozkazów o adresach parzystych i nieparzystych rezydują w odrębnych blokach pamięci ROM o organizacji 32 K, dołączonych do 8 linii mniej i bardziej znaczącej części systemowej magistrali danych. Dostęp do modułu pamięci ROM osiąga się od adresu 0F0000H (pierwszej 1–megabajtowej przestrzeni adresowej) i FF0000H (ostatniej 1–megabajtowej przestrzeni adresowej). Cykl odczytu pamięci ROM wymaga 2 lub więcej dodatkowych taktów, (Proc CLK) zależnie od czasu dostępu zastosowanych pamięci i od częstotliwości pracy mikroprocesora. W tym drugim przypadku cykl wydłuża się o dodatkowe takty oczekiwania T_w (zwykle jeden), sygnałem $\overline{ARDY}$ z układu generowania stanów WAIT.

3.2.3. Podsystem pamięci RAM

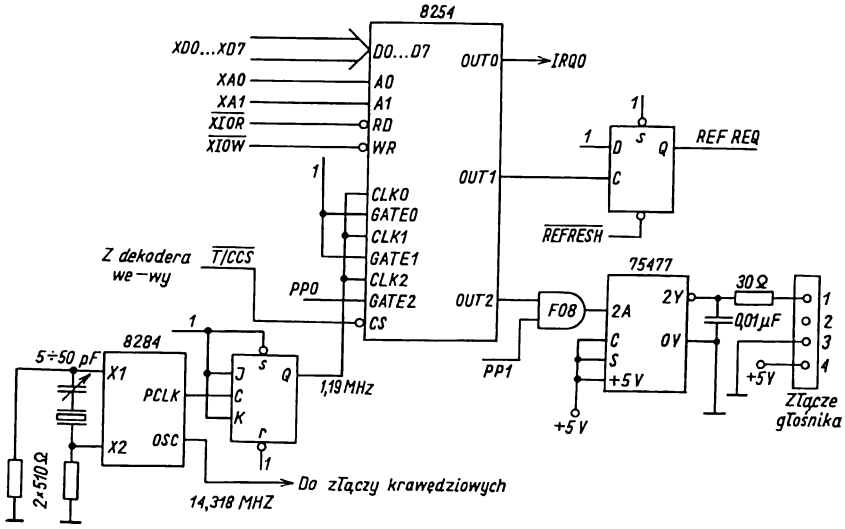
Pamięć RAM jest pamięcią dynamiczną, zbudowaną z 4 bloków o organizacji 128 K × 8b – rys. 3.18. Dwa z tych bloków dołączono do mniej znaczącej części, a pozostałe dwa – do bardziej znaczącej części systemowej magistrali danych przez szybkie 2–kierunkowe bufory (74ALS245). Bloki te, podobnie jak w PC/XT, są zbudowane z układów pamięci DRAM o organizacji 64 K × 1, 256 K ×1 (wtedy



Rys. 3.18. Schemat blokowy pamięci RAM w IBM PC/AT

występują tylko dwa takie bloki) lub 1 M × 1. Adres wiersza i kolumny jest generowany na wyjściach szybkich multiplekserów 74F158, na podstawie odpowiedniej liczby mniej

znaczących bitów adresowych MA0, MA1, ..., a sygnały $\overline{RAS}_i$, $\overline{CAS}_L$ i $\overline{CAS}_H$ są generowane w układzie dekodera pamięci (ROM, RAM), na podstawie bardziej znaczących bitów adresowych A17...A23.



Rys. 3.19. Schemat blokowy programowanego układu czasowego 8254

Tablica 3.6

Adres szesnastkowy	Nazwa i przeznaczenie bloku
000000 ... 07FFFF	512 KB systemowej pamięci RAM
080000 ... 09FFFF	128 KB rozszerzenia pamięci RAM, dołączonej przez złącza krawędziowe
0A0000 ... 0BFFFF	128 KB pamięci obrazu (ang. *video* RAM)
0C0000 ... 0DFFFF	128 KB pamięci ROM na kartach we-wy
0E0000 ... 0EFFFF	64 KB rezerwowej pamięci ROM na płycie głównej (zdublowanej od adresu FE0000H)
0F0000 ... 0FFFFF	64 KB pamięci ROM na płycie głównej (zdublowanej od adresu FF0000H)
100000 ... FDFFFF	15 MB rozszerzenia pamięci RAM, dołączonej przez złącza krawędziowe
FE0000 ... FEFFFF	64 KB rezerwowej pamięci ROM na płycie głównej, dublującej pamięć ROM od adresu 0E0000H
FF0000 ... FFFFFF	64 KB pamięci ROM na płycie głównej, dublującej pamięć ROM od adresu FFFFFFH

Pamięć RAM umieszczono w początkowym obszarze adresowania, od adresu 0000000H w 16-megabajtowej przestrzeni adresowej. Układ odświeżania wysyła żądanie cykli odświeżania pamięci co 15,09 μs. W jednym cyklu jest odświeżany jeden określony wiersz we wszystkich blokach pamięci dynamicznej. Sygnał żądania odświe-

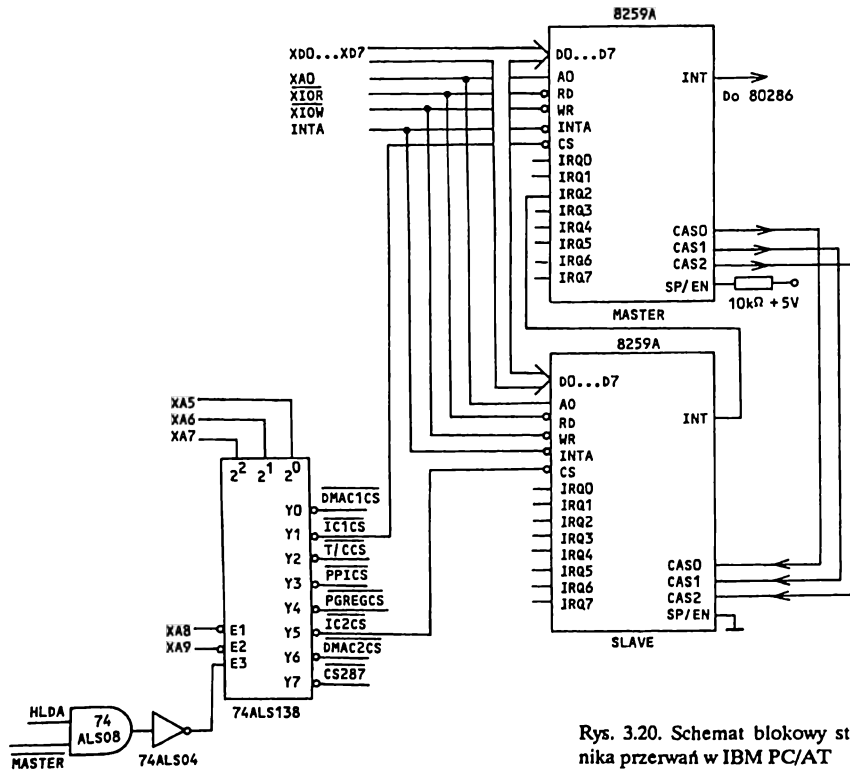
żania jest generowany na wyjściu kanału 1 (OUT1) *programowanego układu czasowego* (8254) – rys. 3.19. W programie, w którym inicjuje się pamięć RAM, steruje się (między innymi) kanał 1 układu czasowego tak, by na jego wyjściu pojawiał się sygnał żądania odświeżania co 15,09 ms (okres ten można zmieniać programowo).

Układ kontroli parzystości w pamięci DRAM rozwiązano podobnie jak w mikrokomputerze PC/XT.

Pamięć RAM można rozszerzyć o dodatkowe bloki, umieszczone na kartach dołączonych przez złącza krawiędziowe. Produkowane w ostatnich latach mikrokomputery IBM PC/AT zawierają na płycie głównej zwykle pamięć RAM o pojemności 1MB. Przeznaczenie 16–megabajtowej przestrzeni adresowej pamięci ilustruje tabl. 3.6.

3.2.4. Podsystem we–wy

Układy we–wy na płycie głównej dołączono do magistrali zewnętrznej (zbuforowanej magistrali systemowej). Są one wybierane sygnałami z dekodera wewnętrznych układów we–wy (74ALS138) – rys. 3.20.



Rys. 3.20. Schemat blokowy sterownika przerwań w IBM PC/AT

Przeznaczenie przestrzeni adresowej we–wy przedstawiono w tabl. 3.7. Układy we–wy, znajdujące się na płycie głównej, korzystają tylko z adresów od 000H do 0FFH. Do układów tych należy m.in. *programowany układ czasowy* 8254 – w którym są realizowane zależności czasowe, niezbędne do prawidłowej pracy mikrokomputera.

Tablica 3.7

Adres szesnastkowy	Urządzenie
000 ... 01F	Sterownik DMA nr 1 (8237A - 5)
020 ... 03F	Sterownik przerwań nr 1 (8259A) - MASTER
040 ... 05F	Układ czasowy (8254)
060 ... 06F	Sterownik klawiatury (8042)
070 ... 07F	Zegar czasu rzeczywistego, maska NMI
080 ... 09F	Rejestr stron DMA (74LS612)
0A0 ... 0BF	Sterownik przerwań nr 2 (8259A) - SLAVE
0C0 ... 0DF	Sterownik DMA nr 2 (8237A - 5)
0F0	Zerowanie BUsy koprocesora arytmetycznego
0F1	Zerowanie koprocesora arytmetycznego
0F8 ... 0FF	Koprocesor arytmetyczny
1F0 ... 1F8	Dysk stały
200 ... 207	We-wy "Game"
278 ... 27F	We-wy rwnoległe nr 2 (drukarki)
2F8 ... 2FF	We-wy szeregowe nr 2
300 ... 31F	Karta prototypu
360 ... 36F	Zarezerwowane
378 ... 37F	We-wy równoległe nr 1 (drukarki)
380 ... 38F	SDLC, bitowosynchroniczna nr 2
3A0 ... 3AF	Bitowosynchroniczna nr 1
3B0 ... 3BF	Monitor monochromatyczny
3C0 ... 3CF	Zarezerwowane
3D0 ... 3DF	Moduł grafiki kolorowej
3F0 ... 3F7	Sterownik dysków elastycznych
3F8 ... 3FF	We-wy szeregowe nr 1

Układ czasowy 8254 ma trzy niezależne kanały 0, 1 i 2. Na wejścia zliczające tych kanałów są wprowadzone impulsy generowane na wyjściu PCLK układu 8284, których częstotliwość jest dwukrotnie zmniejszona w układzie dzielnika, zbudowanego z przerzutnika 74LS112 (rys. 3.19). Kanał 0 pełni funkcję systemowego układu czasowego, który generuje na wyjściu OUT0 przerwanie IRQ0, kierowane do nadrzędnego stero-wnika przerwań nr 1 (IC1 – ang. *interrupt controller* 1). Kanał 1 pełni funkcję układu generowania sygnału żądania odświeżania dynamicznej pamięci RAM. Programuje się go tak, aby cyklicznie (zwykle co 15,09 μs) generował sygnał żądania odświeżania pamięci DRAM. Zmieniając stałą czasową licznika tego kanału, można zmienić częstotliwość sygnałów na wyjściu OUT1. Kanał 2 wykorzystuje się do generacji sygnału dźwiękowego. Obwód generacji sygnału dźwiękowego rozwiązano podobnie, jak w IBM PC/XT. Wejście bramkujące tego kanału steruje się w programie ustawieniem stanu 0 lub 1 na linii PP0 wyjścia (portu) równoległego (74ALS175) w układzie sterownika klawiatury. Adres tego portu jest równy 061H. Sygnał wyjściowy OUT2,

przed wprowadzeniem na głośnik, jest bramkowany sygnałem PP1 z tego samego wyjścia równoległego.

Innym układem we–wy jest *sterownik przerwań*, utworzony z dwóch układów 8259A (IC1 i IC2), połączonych kaskadowo. IC2 jest układem podrzędnym, a IC1 – nadrzędnym (patrz rys. 3.20). Część wejść przerywających ma stałe przeznaczenie, pozostałe są do dyspozycji użytkownika.

W sterowniku nadrzędnym na wejścia żądania przerwania IRQ_i (i = 0...7) doprowadza się:

IRQ0 – przerwanie z wyjścia OUT0 układu 8254;
IRQ1 – przerwanie ze sterownika klawiatury – po zapełnieniu bufora wyjściowego (po skompletowaniu kodu przycisku);
IRQ2 – przerwanie ze sterownika podrzędnego IC2;
IRQ3 – przerwanie z układu we–wy szeregowego nr 2;
IRQ4 – przerwanie z układu we–wy szeregowego nr 1;
IRQ5 – przerwanie z układu we–wy równoległego nr 2;
IRQ6 – przerwanie ze sterownika dysków elastycznych;
IRQ7 – przerwanie ze sterownika we–wy równoległego nr 1.

W sterowniku podrzędnym na wejścia żądania przerwania IRQ doprowadza się:

IRQ0 (IRQ8) – przerwanie z zegara czasu rzeczywistego;
IRQ1 (IRQ9) – przesunięte programowo (INT 0AH) do IRQ2 w sterowniku nadrzędnym;
IRQ5 (IRQ13) – przerwanie z koprocesora arytmetycznego;
IRQ6 (IRQ14) – przerwanie ze sterownika dysków sztywnych.

Pozostałe wejścia IRQ2 (IRQ10), IRQ3 (IRQ11), IRQ4 (IRQ12), IRQ7 (IRQ15) są do dyspozycji użytkownika.

Sterownik DMA, w mikrokomputerze IBM PC/AT, zbudowano z dwóch układów 8237A (DMAC1 i DMAC2 – ang. *direct memory access controller*), połączonych kaskadowo, tworząc 7 kanałów bezpośredniego dostępu do pamięci (rys. 3.21). Funkcje realizowane za pośrednictwem kanałów K0...K3 w sterowniku podrzędnym DMAC1 są następujące:

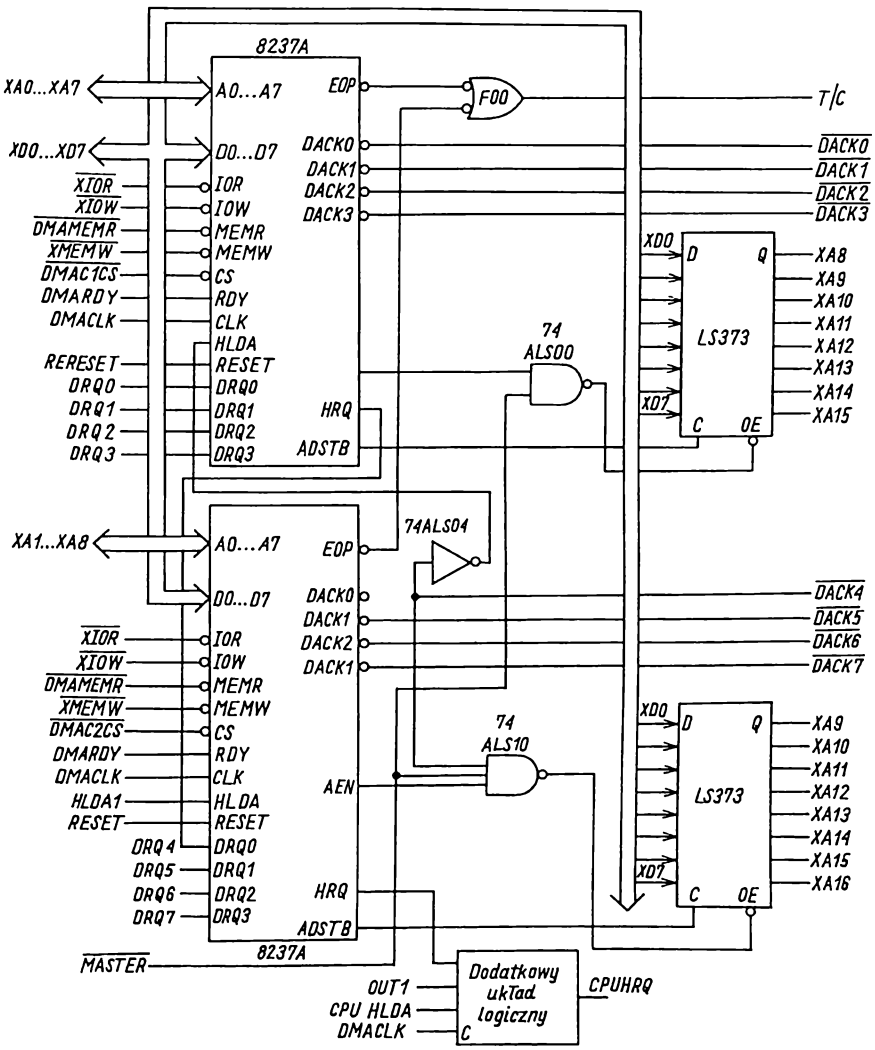
K1 – przesłania zgodne z protokołem SDLC;
K2 – przesłania między pamięcią dyskową (dyski elastyczne) a pamięcią operacyjną;
K0 i K3 – są do dyspozycji użytkownika.

W sterowniku DMAC2 (nadrzędnym) kanały K1 (K5), K2 (K6), K3 (K7) są do dyspozycji użytkownika, a do kanału K0 (K4) dołączono sterownik podrzędny DMAC1.

Kanały sterownika podrzędnego umożliwiają przesłania 8–bitowych danych między 8–bitowymi układami we–wy i 8 lub 16–bitowym blokiem pamięci dynamicznej. Dane 8–bitowe przesyła się blokami o pojemności mniejszej lub równej 64 KB w ramach całej 16–megabajtowej przestrzeni adresowej pamięci.

Kanały 1, 2, 3 sterownika nadrzędnego umożliwiają przesłania 16–bitowych danych między 16–bitowymi układami we–wy a 16–bitowym blokiem pamięci dynamicznej. Dane 16–bitowe przesyła się blokami o pojemności 128 KB w ramach całej 16–megabajtowej przestrzeni adresowej. Ponieważ przesłania DMA mogą odbywać się między dowolnym obszarem (w obrębie 16 megabajtów) pamięci, a sterownik DMA

(8237A) może generować tylko 16-bitowy adres, jest niezbędne (tak jak w PC XT) zastosowanie rejestrów strony dla poszczególnych kanałów. W mikrokomputerze IBM PC/AT zastosowano do tego celu układ 74LS612. Przy przesłaniach w kanałach 0, 1, 2, 3 sterownika DMAC1, umożliwia on ustawienie bardziej znaczących bitów adresu (A16...A23). 16 mniej znaczących bitów adresu określają rejestry adresowe tych kanałów. Sygnał $\overline{BHE}$ jest w tym przypadku generowany jako negacja sygnału A0 (stąd tylko przesłania danych 8-bitowych).



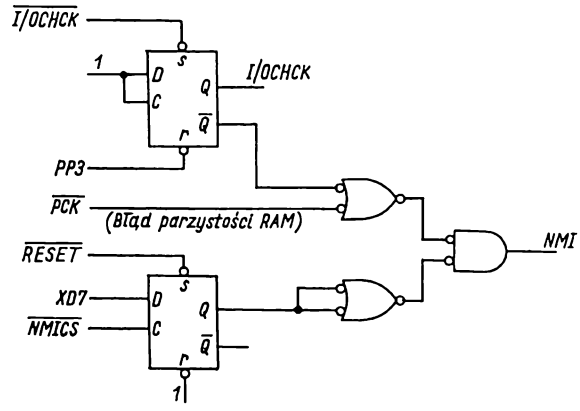
Rys. 3.21. Schemat blokowy sterownika DMA w IBM PC/AT

Przy przesłaniach kanałami 1, 2, 3 (5, 6, 7) sterownika DMAC2, układ rejestrów stron określa bity od A17 do A23, a rejestry adresowe kanałów – bity od A1 do A16; sygnał $\overline{BHE}$ i bit adresowy A0 są ustawione w stan 0 (co zapewnia przesłania danych 16-bitowych w jednym cyklu DMA).

Przyporządkowanie adresów w przestrzeni we-wy rejestrom stron poszczególnych kanałów DMA jest następujące:

087 H – rejestr strony kanału 0;
083 H – rejestr strony kanału 1;
081 H – rejestr strony kanału 2;
082 H – rejestr strony kanału 3;
08B H – rejestr strony kanału 5;
089 H – rejestr strony kanały 6;
08A H – rejestr strony kanału 7.

Układ generowania przerwania niemaskowalnego NMI rozwiązano podobnie, jak w PC/XT. Przerwanie NMI można zewnętrznie zablokować, zapisując słowo 80H lub odblokować, zapisując słowo 00H do przerzutnika maski (74ALS74) wybieranego adresem 070H. Przerwanie niemaskowalne może być generowane, gdy wystąpi błąd parzystości w bloku pamięci umieszczonej na płycie głównej (aktywny sygnał PCK) oraz na modułach rozszerzających (aktywny sygnał $\overline{I/OCHCK}$ – ang. *input/output channel check*). Przedstawiono to na rys. 3.22. Przerzutnik $\overline{S}\,\overline{R}$ w torze $\overline{I/OCHCK}$ zeruje się ustawiając bit PP3 = 0 układu we-wy, wybieranego adresem 061 H.



Rys. 3.22. Układ generowania przerwania niemaskowanego

Funkcję *układu we-wy klawiatury* pełni *mikrokomputer jednoukładowy* 8042. Jego zadaniem jest odbieranie danych szeregowych (kodów przycisków) z klawiatury, sprawdzenie bitu parzystości, konwersja tych danych ma postać równoległą i ich wyprowadzenie na magistralę danych. Złącze służące do przyłączenia klawiatury jest takie samo, jak w IBM PC/XT. Układ we-wy umożliwia również przesyłanie danych szeregowych (z bitem parzystości) do układu sterownika klawiatury, w przypadku wykrycia błędu parzystości przy odbiorze kodu znaku z klawiatury.

Jak już wspomniano, w mikrokomputerze PC/AT znajduje się również *zegar czasu rzeczywistego.* Funkcję tę pełni układ MC 146818, wykonany w technologii CMOS, z dodatkowym zasilaniem bateryjnym. Zawiera on zegar czasu rzeczywistego i 64–bajtową pamięć RAM CMOS. Układ zegara wykorzystuje 14 bajtów tej pamięci, a pozostałe służą do określenia konfiguracji systemu.

Przeznaczenie bajtów pamięci określono w tabl. 3.8, a informacje niezbędne do prawidłowej pracy układu zegara przechowywane w tej pamięci przedstawiono w tabl. 3.9.

Tablica 3.8

Adres szesnastkowy	Przeznaczenie
00 ... 0D	Dane dla układu zegara czasu rzeczywistego
0E	Stan diagnostyczny
0F	Stan wstrzymania
10	Rodzaj napędu dysków elastycznych (A i B)
11	Niewykorzystany
12	Rodzaj dysku stałego (C i D)
13	Niewykorzystany
14	Bajt wyposażenia
15	Mniej znaczący bajt konfiguracji bloku pamięci podstawowej
16	Bardziej znaczący bajt konfiguracji bloku pamięci podstawowej
17	Mniej znaczący bajt konfiguracji bloku rozszerzania pamięci
19	Bardziej znaczący bajt konfiguracji bloku rozszerzania pamięci
19 ... 20	Niewykorzystany
2E ... 2F	Dwubajtowa suma kontrolna pamięci CMOS
30	Mniej znaczący bajt konfiguracji bloku rozszerzania pamięci
31	Bardziej znaczący bajt konfiguracji bloku rozszerzania pamięci
32	Bajt daty
33	Znaczniki danych ustawiane podczas włączania napięcia
34 ... 3F	Niewykorzystane

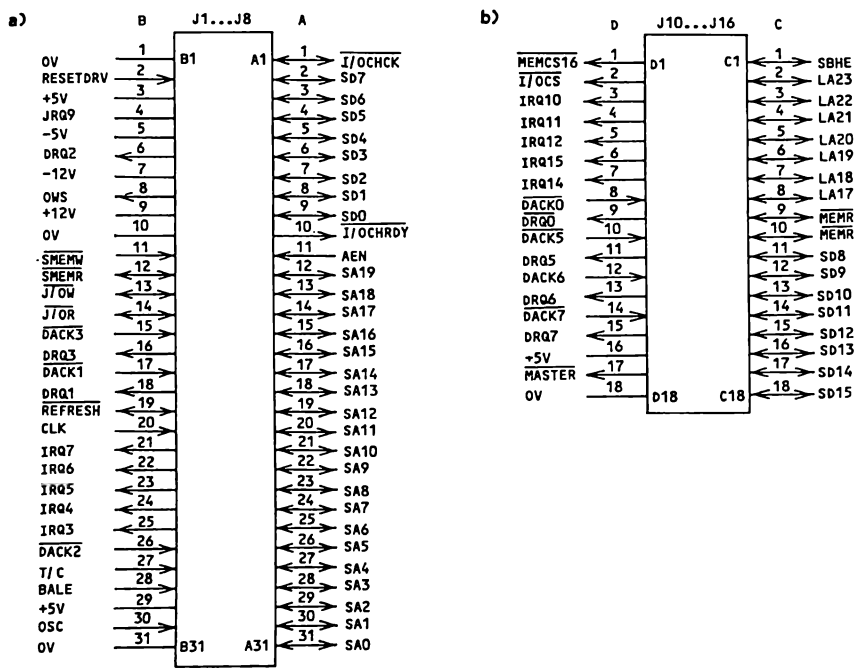
Zapis do pamięci CMOS realizuje się w dwu krokach: w pierwszym podaje się adres komórki pamięci (zapis układu we–wy o adresie 70H), a w drugim zapisuje się daną do wybranej komórki (zapis układu we–wy o adresie 71H). Podobnie, w dwu krokach, wykonuje sie operację odczytu: podaje się adres odczytywanej komórki (odczyt do układu we–wy o adresie 70H), a w drugim odczytuje (odczyt układu we–wy o adresie 71H).

Złącza krawędziowe (kanały we–wy), znajdujące się na płycie głównej, dołączono do magistrali systemowej. Typowy mikrokomputer IBM PC/AT wyposażono w 8 złączy krawędziowych 62–stykowych (J1...J8) i 6 złączy krawędziowych 36–stykowych (J10...J16). Umożliwiają one rozszerzenie systemu o dodatkowe układy we–wy 8– lub 16–bitowe oraz o pamięci (doprowadzany jest do nich pełny 24–bitowy adres

i odpowiednie sygnały sterujące). Wszystkie linie sygnałowe dołączone do złączy krawędziowych są zgodne z poziomami TTL. Dołączony układ we–wy nie powinien wnosić większego obciążenia niż 2 układy 74LS na jedną linię sygnałową. Przyporządkowanie sygnałów złączom J1...J8 i J10...J16 przedstawiono na rys. 3.23.

Tablica 3.9

Adres szesnastkowy	Informacje
00	Sekundy
01	Alarm (sekund)
02	Minuty
03	Alarm (minut)
04	Godziny
05	Alarm (godzin)
06	Dzień tygodnia
07	Dzień miesiąca
08	Miesiąc
09	Rok
0A	Rejestr stanu A
0B	Rejestr stanu B
0C	Rejestr stanu C
0D	Rejestr stanu D



Rys. 3.23. Przyporządkowanie sygnałów złączom krawędziowym w IBM PC/AT: a) 62-stykowym; b) 36-stykowym

Sygnały SA0...SA19 są używane do adresowania pamięci i układów we–wy systemu (uzupełnione sygnałami LA17...LA23, umożliwiają adresowanie w całej 16–megabajtowej przestrzeni adresowej). Sygnały SA0...SA19 są doprowadzane do magistrali systemowej z rejestrów adresowych, sterowanych buforowanym sygnałem ALE (BALE). Opadające zbocze tego sygnału powoduje zapamiętanie adresu w tych rejestrach.

Sygnały LA17...LA23 nie są zapamiętywane w rejestrach; są one ważne tylko w czasie trwania aktywnego stanu sygnału BALE i stanowią zwykle sygnały wejściowe dekoderów pamięci. Sygnały wyjściowe z tych dekoderów mogą być zapamiętywane w rejestrach układów we–wy na opadającym zboczu sygnału BALE. Źródłem tych sygnałów może być mikroprocesor 80286 lub sterownik DMA znajdujący się na płycie głównej albo mikroprocesor lub sterownik DMA znajdujący się na modułach rozszerzających, dołączonych przez złącza krawędziowe.

Sygnał $\overline{\text{I/OCHCK}}$ wskazuje, poziomem wysokim, wystąpienie błędu parzystości przy odczytywaniu pamięci lub urządzenia w kanale we–wy.

Sygnał gotowości kanału $\overline{\text{I/OCHRDY}}$ umożliwia, poziomem niskim, wydłużenie cyklu magistrali dla wolnych pamięci lub układów we–wy, znajdujących się w kanałach we–wy.

Sygnały $\overline{\text{IOR}}$ i $\overline{\text{IOW}}$ są sygnałami odpowiednio odczytu i zapisu układów we–wy. Sygnał odczytu pamięci $\overline{\text{SMEMR}}$ pojawia się tylko przy odwołaniach do pamięci umie-szczonych w początkowej 1–megabajtowej przestrzeni adresowej, sygnał $\overline{\text{MEMR}}$ jest aktywny we wszystkich cyklach odczytu pamięci. Sygnał $\overline{\text{SMEMR}}$ jest generowany na podstawie sygnału wyboru początkowej 1–megabajtowej przestrzeni adresowej i sygnału $\overline{\text{MEMR}}$. Źródłem tych sygnałów może być mikroprocesor lub sterownik DMA znaj-dujący się na płycie głównej lub w kanale we–wy. W tym drugim przypadku adres powinien być ustalony na 1 okres zegara systemowego przed wysłaniem sygnału $\overline{\text{MEMR}}$. Takie same uwagi odnoszą się do sygnałów zapisu $\overline{\text{MEMW}}$ i $\overline{\text{SMEMW}}$.

DRQ0...DRQ3 i DRQ5...DRQ7 to asynchroniczne sygnały żądania bezpośredniego dostępu do pamięci, generowane w układach we–wy rozszerzających konfigurację systemu. Żądanie DRQ0 ma najwyższy, a DRQ7 najniższy priorytet. Układ generujący sygnał $\overline{\text{DRQ}_i}$ powinien go utrzymywać do chwili uzyskania potwierdzenia przejęcia sterowania magistrali, czyli do chwili, gdy sygnał $\overline{\text{DACK}_i} = 0$.

Sygnał AEN zapewnia blokowanie układów w kanale we–wy podczas przesłań DMA. Gdy AEN = 1, źródłem adresów i sygnałów sterujących zapisem, odczytem jest sterownik DMA.

Sygnał $\overline{\text{REFRESH}}$ wskazuje cykl odświeżania pamięci.

Sygnał T/C pojawia się po zakończeniu przesłania bloku danych w każdym z kanałów DMA. Sygnał $\overline{\text{SBHE}}$ umożliwia przesłanie danych liniami SD8...SD15 (odblokowuje bufory trójstanowe).

Sygnały $\overline{\text{MASTER}}$ i DRQ_i umożliwiają mikroprocesorowi lub sterownikowi DMA znajdującemu się w kanale we–wy przejęcie sterowania magistrali systemowej. Po przyjęciu sygnału $\overline{\text{DACK}_i}$ sterownik DMA lub mikroprocesor ustawia sygnał $\overline{\text{MASTER}} = 0$, co powoduje między innymi zablokowanie dekodera układów we–wy płyty głównej i przejęcie sterowania magistrali (odłączenie innych źródeł sygnałów

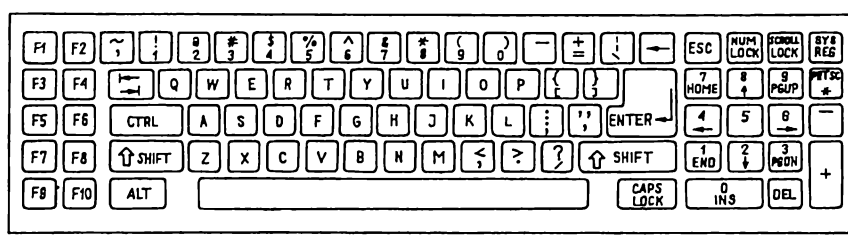
magistrali systemowej). Sygnał ten nie może trwać dłużej niż 15 μs, gdyż groziłoby to zakłóceniem cykli odświeżania pamięci. Sygnał $\overline{\text{MEMCS16}}$ = 0 wskazuje, że bieżące przesłanie danych jest 16-bitowe i odbywa się w cyklu pamięci z 1 taktem oczekiwania T_w. Podobnie, sygnał $\overline{\text{I/OCS16}}$ sygnalizuje, że bieżące przesłanie danych jest 16-bitowe i odbywa się w cyklu we-wy z 1 taktem oczekiwania. Sygnały te, otrzymywane na podstawie stanu linii adresowych, powinny być doprowadzane z nadajników typu OC (otwarty kolektor) lub trójstanowych do złącza krawędziowego.

OSC jest sygnałem impulsów zegarowych o częstotliwości 14,31818 MHz doprowadzonym do złącza z układu 8284 znajdującego się na płycie głównej.

Sygnał $\overline{\text{OWS}}$ (generowany w kanałach we-wy) określa, że mikroprocesor może zakończyć cykl magistrali bez wtrącania dodatkowych taktów oczekiwania. Sygnał ten należy doprowadzić z nadajnika typu OC lub trójstanowego do złącza krawędziowego. Sygnały CLK, RESET DRV, $\overline{\text{IOR}}$, $\overline{\text{IOW}}$, SD0...SD15 oraz linie zasilania były dokładnie opisane. Pozostałe złącza standardowe – klawiatury, zasilacza, głośnika – są identyczne z odpowiednikami w mikrokomputerze IBM PC/XT.

3.3. Moduł klawiatury

Klawiatura umożliwia użytkownikowi bezpośrednie wprowadzenie informacji do mikrokomputera. Klawiatura mikrokomputerów IBM PC jest oddzielną jednostką zawierającą własny procesor i łącze szeregowe, umożliwiające komunikację z modułem głównym mikrokomputera. Stosuje się różne typy klawiatur, o różnej liczbie przycisków (od 83 do 101) i różnym ich rozmieszczeniu. Typowa klawiatura ma 84 przyciski (rys. 3.24). Istnieje kilkanaście narodowych wersji klawiatur, zawierających przyciski ze znakami specyficznymi dla danego języka narodowego (np. wersja amerykańska, angielska, francuska, włoska, niemiecka, hiszpańska), ale brak w tym zestawie klawiatury polskiej.



Rys. 3.24. Typowa klawiatura PC z 84 przyciskami

Klawiatura ma zasadniczo trzy grupy przycisków
- alfanumeryczne,
- sterujące,
- funkcyjne.

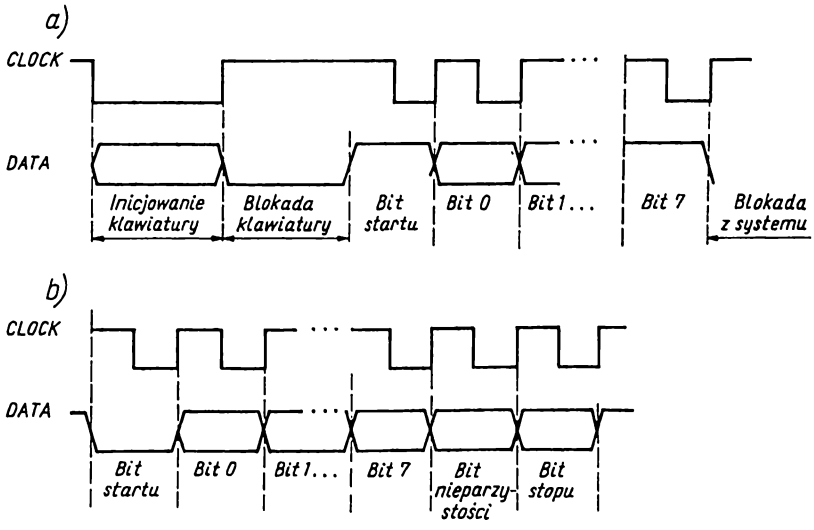
Konstrukcja układowa modułu klawiatury jest stosunkowo prosta. Część zasadniczą stanowi mikrokomputer jednoukładowy (np. 8042, 8048, 8049), spełniający

funkcję sterownika klawiatury. Sterownik służy między innymi do: wykonywania autotestu po włączeniu zasilania lub na żądanie systemu; sterowania transmisją między klawiaturą a systemem; przeglądania stanu przycisków (przy zastosowaniu układu 74LS138); sterowania diodami świecącymi, obrazującymi stan klawiatury lub mikrokomputera (np. CAPS LOCK, NUM LOCK, SCROLL LOCK, POWER ON).

3B/05	3C/06	29/0E	02/16	03/1E	04/26	05/25	06/2E	07/36	08/3D	09/3E	0A/46	0B/45	0C/4E	0D/55	2B/5D	0E/66	01/76	45/77	46/7E	/84
3D/04	3E/0C	0F/0D	10/15	11/1D	12/24	13/2D	14/2C	15/35	16/3C	17/43	18/44	19/4D	1A/54	1B/5B			47/6C	48/75	49/7D	37/7C
3F/03	40/0B		1D/14	1E/1C	1F/1B	20/23	21/2B	22/34	23/33	24/3B	25/42	26/4B	27/4C	28/52		1C/5A	4B/6B	4C/73	4D/74	4A/7B
41/83	42/0A		2A/12		2C/1A	20/22	2E/21	2F/2A	30/32	31/31	32/3A	33/41	34/49	35/4A		36/59	4F/69	50/72	51/74	4E/79
43/01	44/09	38/11						39/29								3A/58		52/70	53/71	

Rys. 3.25. Kody przycisków klawiatury w $\frac{\text{PC/XT}}{\text{PC/AT}}$

Każdemu przyciskowi klawiatury przyporządkowano określony kod 8-bitowy. Do mikrokomputera są przesyłane kody odpowiadające przyciskom, a nie standardowe kody przypisanych im znaków. Przy naciśnięciu przycisku są wysyłane odpowiednie kody, które zestawiono na rys. 3.25.



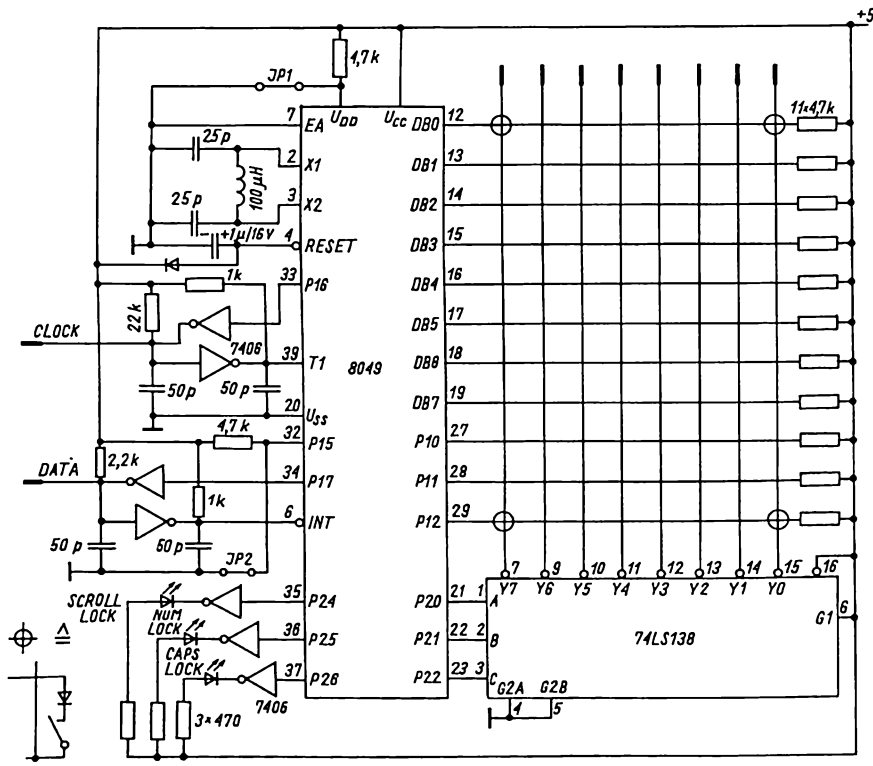
Rys. 3.26. Przebieg transmisji danych w mikrokomputerach: a) PC/XT; b) PC/AT

W klawiaturze PC/XT przy zwalnianiu danego przycisku jest tworzony inny kod przez dodanie wartości 80H do kodu generowanego przy naciśnięciu. W klawiaturze PC/AT przy zwalnianiu przycisku jest przesyłany kod w postaci dwóch bajtów – kodu naciśnięcia, poprzedzonego kodem F0H.

Transmisję między klawiaturą a mikrokomputerem realizuje się za pomocą dwóch linii: CLOCK (linia zegara) i DATA (linia danych).

Informacje przesyłane z klawiatury mają różne postaci w mikrokomputerach PC/XT (rys. 3.26a) i PC/AT (rys. 3.26b). Podczas transmisji do mikrokomputera klawiatura generuje impulsy na linii CLOCK, które taktują przesyłaniem kolejnych bitów. Mikrokomputer może wstrzymać wysyłanie danych z klawiatury, wymuszając na linii CLOCK stan niski.

W mikrokomputerze PC/AT transmisję realizuje się w obydwu kierunkach. Przykładowe rozwiązanie modułu klawiatury przedstawiono na rys. 3.27.



Rys. 3.27. Schemat logiczny przykładowego rozwiązania modułu klawiatury

Naciśnięcie lub zwolnienie przycisku klawiatury jest zgłaszane do systemu BIOS jako przerwanie, informujące o działaniu klawiatury. Tłumaczenie kodów prze-

szukiwania jest trudne, ponieważ działanie klawiatury zmienia się po naciśnięciu przycisków: Shift, Alt, Ctrl. Na przełączenia klawiatury oddziałują również przyciski Caps Lock i Num Lock.

Kod przeszukiwania jest tłumaczony na dwa bajty (mniej znaczący – wartość ASCII, odpowiadająca danemu przyciskowi; bardziej znaczący – kod przycisku), a następnie przechowywany w określonym obszarze pamięci (0000...041EH).

Podczas tłumaczenia kodów przeszukiwania są sprawdzane również możliwości kombinacji przycisków, takie jak np.:

- Ctrl + Num Lock – zawieszenie (ang. *pause*) obsługi programu, aż do pojawienia się innego znaku;
- Ctrl + Scroll Lock – wstrzymywanie (ang. *break*) działania wykonywanego programu i identyfikacja linii przerwania;
- Ctrl + Alt + Del – ponowny start systemu operacyjnego (ang. *system reset*);
- Shift + PrtSc – wydrukowanie treści ekranu na drukarce (ang. *print screen*);
- Ctrl + PrtSc – echo ekranu na drukarce. Każda linia jest drukowana na drukarce w takiej postaci, w jakiej wprowadza ją użytkownik;
- Ctrl + Home – usunięcie informacji na ekranie i przesunięcie kursora do pozycji wejściowej w lewym górnym rogu ekranu. Funkcje przycisków funkcyjnych mogą być różne w zależności od wykonywanego programu użytkowego.

3.4. Moduły we–wy

Układy pośredniczące między magistralą systemu mikroprocesorowego a szeroko pojętym jego otoczeniem przyjęło się nazywać *układami we–wy* lub *układami sprzęgającymi* (ang. *interface*). Uwzględniając rodzaje występujących sygnałów, można wśród układów we–wy rozróżnić:

- układy wejść analogowych;
- układy wyjść analogowych;
- układy wejść cyfrowych;
- układy wyjść cyfrowych.

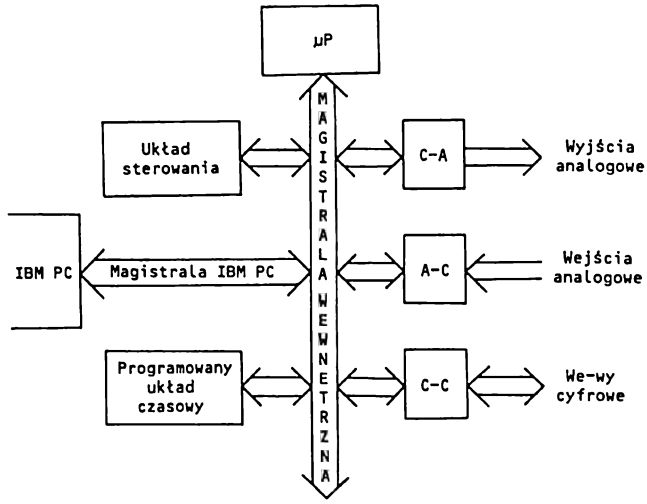
Układy we–wy zrealizowane w określonym standardzie logicznym, elektrycznym i mechanicznym, są nazywane *modułami we–wy*.

Wzajemne przekazywanie sygnałów analogowych i cyfrowych odbywa się przy zastosowaniu przetworników analogowo–cyfrowych (a–c) i cyfrowo–analogowych (c–a).

Dwukierunkowe przesyłanie sygnałów cyfrowych między systemami a otoczeniem, można zrealizować szeregowo lub równolegle, w sposób asynchroniczny lub synchroniczny, stosując typowe układy scalone np. 8250, 8251, 8255, 8253 i 8254.

Uproszczony schemat blokowy wielofunkcyjnyego modułu we–wy przedstawiono na rys. 3.28.

Specjalizowany mikroprocesor modułu spełnia funkcje sterownika, organizującego wymianę informacji między modułem a magistralą IBM PC, odciążając procesor główny mikrokomputera.



Rys. 3.28. Schemat blokowy wielofunkcyjnego modułu we-wy

Operacje we-wy można sterować za pomocą określonej liczby rejestrów, umieszczonych w przestrzeni adresowej mikroprocesora przeznaczonej dla we-wy. Rejestry te, w mikroprocesorze IBM PC, mogą mieć sąsiednie adresy w jego przestrzeni adresowej dla urządzeń we-wy, jeśli rejestry rozkazów i statusu są umieszczone pod tym samym adresem oraz rejestry danych wejściowych i danych wyjściowych są też pod wspólnym adresem. Adres ten, (dla modułu we-wy) można określić, ustawiając zworki; pod adresem tym są "widziane" rejestry danych.

Wpisaniem nowego bajtu do rejestru rozkazów uaktywnia się mikroprocesor modułu do rozpoczęcia wykonywania wskazanego rozkazu, sterującego działaniem elementów modułu. Kod operacji podstawowych jest zapisywany na pewnej liczbie bitów rejestru rozkazów, pozostałe bity są wykorzystywane do modyfikacji operacji podstawowych. Zasadniczo, moduł wykonuje instrukcje pojedyncze (operujące na jednym bajcie) lub instrukcje blokowe (operujące na blokach danych lub parametrów). Instrukcje blokowe mogą być wykonywane w różnych trybach np. *standardowym blokowym*, *ciągłym blokowym*, *blokowym* z DMA, *ciągłym blokowym* z DMA.

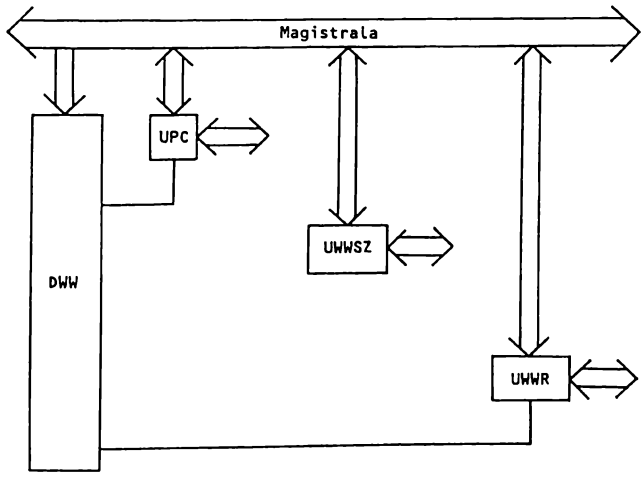
Do taktowania pracy przetworników a-c i c-a stosuje się programowany układ czasowy.

Dostęp do wielofunkcyjnego modułu we-wy jest prosty przy zastosowaniu dowolnego języka programowania, którego interpretator lub kompilator jest do dyspozycji w mikrokomputerze IBM PC. Na przykład, trzy instrukcje języka Basic: *INP*, *OUT* oraz *WAIT* umożliwiają użytkownikowi dostęp do wszystkich funkcji modułu.

Wybrane kody i znaczenie rozkazów podstawowych elementów modułu zestawiono w tabl. 3.10.

Tablica 3.10

Kod rozkazu	Typ		Znaczenie
	pojedynczy	blokowy	
			Rozkazy ogólne
00000000	+		Zerowanie ogólne
00000001	+		Zerowanie błędu
00000010	+		Odczyt zawartości rejestru błędu
00000011	+		Ustawienie okresu zegara wewn.
00001011	+		Testowanie
00001111	+		Stop
			Rozkazy we-wy cyfrowe
00000100	+		Ustawienie portu dla wejść
00000101	+		Ustawienie portu dla wyjść
00000110	+		Natychmiastowy odczyt wejść
00000111	+		Natychmiastowy zapis wyjść
			Rozkazy C-A
00001000	+		Natychmiastowy zapis C-A
00001001		+	Ustawienie parametrów C-A
00001010		+	Zapis C-A
00101010		+	Zapis C-A w trybie ciągłym
00011010		+	Zapis C-A, DMA
			Rozkazy A-C
00001100	+		Natychmiastowy odczyt A-C
00001101		+	Ustawienie parametrów A-C
00001110		+	Odczyt A-C
00101110		+	Odczyt A-C w trybie ciągłym
00011110		+	Odczyt A-C, DMA

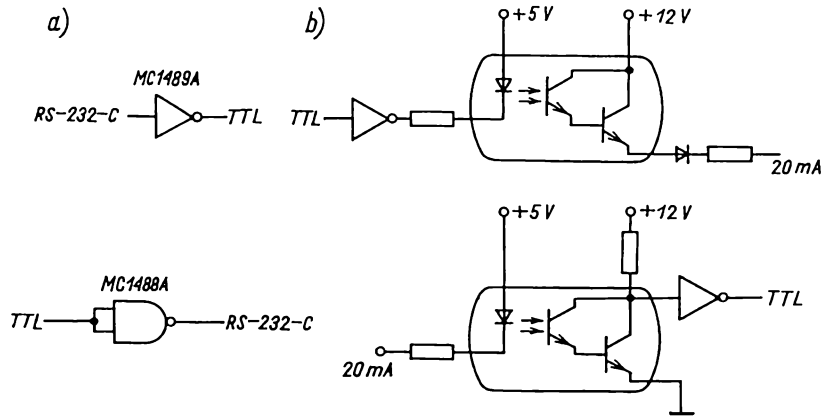


Rys. 3.29. Schemat blokowy standardowego modułu we–wy

Na rysunku 3.29 przedstawiono schemat blokowy standardowego modułu we-wy, zawierającego:
- dekoder we-wy (DWW);
- układ programowania czasu (UPC);
- układ we-wy do transmisji szeregowej (UWWSZ);
- układ we-wy do transmisji równoległej (UWWR).

Dekoder we-wy generuje sygnały selekcji poszczególnych układów składowych modułu. *Układ programowania czasu* zrealizowano przy użyciu układu 8253 lub 8254, którego funkcje definiuje się w programie. Funkcje *układu we-wy do transmisji szeregowej* spełnia układ 8251, a *układu we-wy do transmisji równoległej* - układ 8255.

Opisane moduły są potwierdzeniem współczesnych tendencji w rozwiązaniach układów we-wy systemów mikrokomputerowych. Nadanie modułom cech układów przemysłowych następuje po włączeniu w tory we-wy układów dopasowujących, które zapewniają optoizolację, spełnienie wymagań określonych standardów itp.



Rys. 3.30. Schematy układów translacji: a) TTL ⇆ RS232C; b) TTL ⇆ Pętla 20 mA

Na rysunku 3.30 przedstawiono schematy układów translacji TT L ⇆ RS232C i TTL ⇆ Pętla 20 mA.

3.5. Sterowniki pamięci dyskowych

Pamięci dyskowe, tzw. *dyski elastyczne* (ang. *floppy disk* lub *diskette drive*) lub *stałe* (ang. *hard disk* lub *fixed disk drive*) stanowią podstawowe pamięci zewnętrzne w mikrokomputerach IBM PC/XT i AT. Z tego względu, wielu producentów oferuje moduły sterowników pamięci dyskowych. W początkowym okresie rozwoju mikrokomputerów IBM PC/XT i AT opracowano odrębne *sterowniki dysków elastycznych* i *stałych*. Każdy z nich miał za podstawę zintegrowany, specjalizowany sterownik i kilka

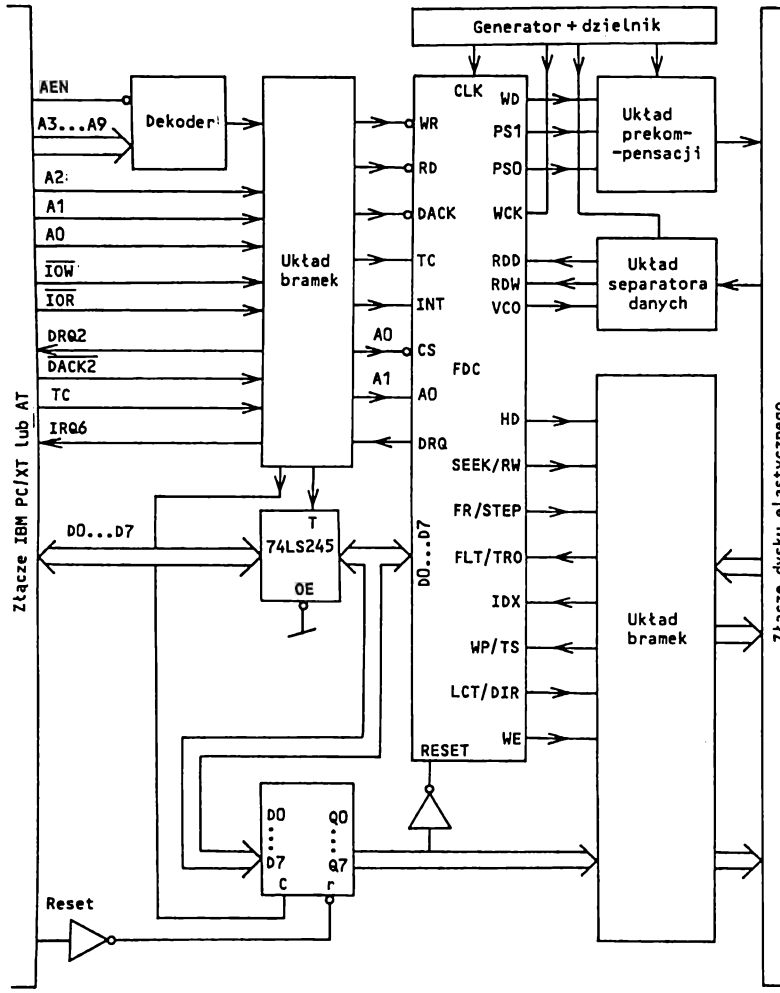
układów scalonych, umożliwiających właściwe sprzęgnięcie sterownika z magistralą mikrokomputera z jednej strony, a z magistralą pamięci dyskowej z drugiej strony. Sterownik taki, jako moduł jednokartowy, łączy się z magistralą systemową mikrokomputera za pomocą złącza krawędziowego, a z jednostką dyskową za pomocą odpowiedniego złącza 34–stykowego.

Do budowy *sterowników dysków elastycznych* stosuje się programowane scalone sterowniki FDC (ang. *floppy disk controller*) np. 8272 (firmy Intel) lub częściej μPD765 (firmy NEC). Umożliwiają one dołączenie do mikrokomputera i obsługiwanie do czterech 40–ścieżkowych jednostek dyskowych (5,25 cala) z zapisem dwustronnym. Zazwyczaj dwie z nich znajdują się wewnątrz obudowy mikrokomputera. Dzięki zastosowaniu zmodyfikowanej modulacji częstotliwości (ang. *modified frequency modulation*), umożliwiają one podwójną gęstość zapisu w formacie IBM 34. Przesyłanie danych między sterownikiem a pamięcią operacyjną odbywa się w trybie bezpośredniego dostępu do pamięci, kanałem 2 sterownika DMA (to znaczy DRQ2 i $\overline{\text{DACK2}}$). Po zakończeniu dowolnej operacji, zainicjowanej odpowiednim poleceniem, sterownik FDC generuje przerwanie IRQ6 do *sterownika przerwań* 8259A. Szybkość przesyłania danych wynosi 250 Kb/s lub 500 Kb/s przy zastosowaniu dysków o zwiększonej pojemności.

Schemat blokowy takiego sterownika przedstawiono na rys. 3.31. Wyróżnić w nim można; sterownik FDC, układ sprzężenia FDC z magistralą mikroprocesora i układ sprzężenia FDC z magistralą jednostki dyskowej. Układ sprzężenia z magistralą mikroprocesora zawiera dwukierunkowy bufor trójstanowy (zwykle 74LS245) na liniach danych, dekoder FDC i rejestr stanowiący układ wyjściowy, do którego zapisuje się dane umożliwiające: wybór jednostki dyskowej (1 z 4), włączenie odpowiedniego silnika i odblokowanie doprowadzania na magistralę systemową sygnałów IRQ6 i DRQ2. Występuje w nim również układ logiczny, zapewniający generowanie sygnałów wyboru sterownika FDC i rejestru wyjściowego oraz odpowiednie uwarunkowania logiczne sygnałów, doprowadzanych z magistrali systemowej do sterownika FDC i sygnałów, wyprowadzanych ze sterownika FDC na magistralę.

Sterownik FDC może wykonywać 15 operacji, wywoływanych odpowiadającymi im poleceniami, takimi jak np. odczytanie danych, zapisanie danych, odczytanie ścieżki, formatowanie ścieżki itd. W programie jest on "traktowany" jako zespół dwóch rejestrów – rejestr danych i główny rejestr stanu, który może być odczytywany w dowolnej chwili przez mikroprocesor. Rejestr danych przyjmuje z mikroprocesora sekwencje bajtów odpowiadające poleceniom i dane przeznaczone do zapisania na dyskietce. Fizycznie rejestr danych jest zbiorem rejestrów tworzących stos, z których w danej chwili tylko jeden ma dostęp do magistrali systemowej. Z tego rejestru procesor pobiera również dane odczytywane z dysku. Zapisywanie danych z podwójną gęstością MFM charakteryzuje się tym, że strumień danych nie zawiera periodycznego ciągu impulsów zegarowych. W związku z tym, przy odczycie danych należy go odtworzyć, w układzie tzw. *separatora danych*, z odczytywanego z dyskietki strumienia informacji. Generowany w tym układzie *sygnał okna danych* RDW (ang. *read window*) umożliwia sterownikowi FDC wydzielenie właściwych danych ze strumienia odczytywanej informacji, wprowadzanego na wejście RDD (ang. *read data*). Separator danych musi śledzić zmiany częstotliwości danych tak, aby przy

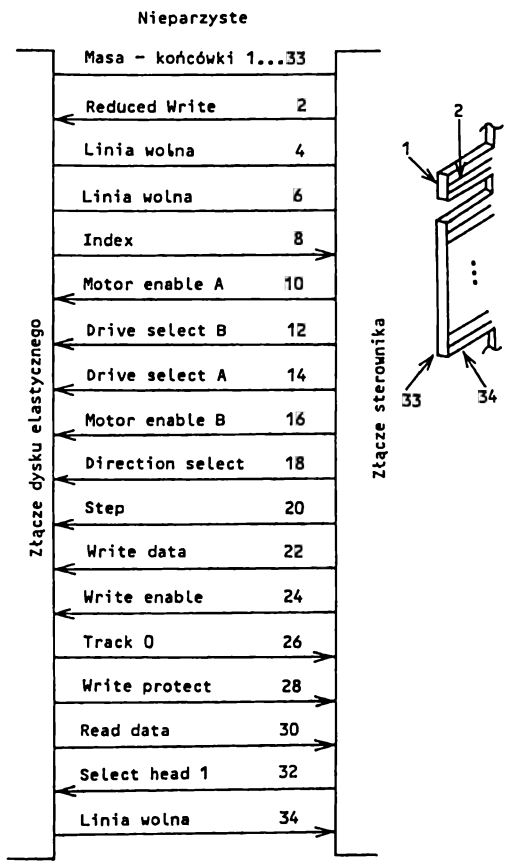
wystąpieniu przesunięcia bitu danych dostroić pozycję sygnału okna, w celu skompensowania tego przesunięcia.



Rys. 3.31. Schemat blokowy sterownika dysków elastycznych

Układ prekompensacji zapisu umożliwia zapisanie danych z podwójną gęstością MFM z wyprzedzeniem (ang. *early*) lub opóźnieniem (ang. *late*), lub bez tych przesunięć względem położenia normalnego (ang. *normal*). Rodzaj przesunięć jest określany automatycznie przez sterownik FDC na podstawie układu bitów w strumieniu danych. Służą do tego sygnały wyjściowe PS1 i PS0. Układ prekompensacji zapisu umożliwia kompensację dających się przewidzieć przesunięć bitów danych

na ścieżce, związanych z przesunięciami maksymalnych wartości prądów w głowicy odczytująco–zapisującej, wynikającymi ze zbyt dużej gęstości zapisu. Kolejne zmiany strumienia magnetycznego występują tak szybko, że pojawiający się w obwodzie głowicy prąd, w wyniku jednej zmiany strumienia, nie zdąży zaniknąć do zera zanim pojawią się kolejne zmiany strumienia. Przesunięcia te, w zależności od zmian kierunku strumienia magnetycznego, powodują, że odczytywane dane mogą się pojawiać wcześniej lub później w stosunku do chwili odpowiadającej pozycji normalnej. Sygnały sterujące, kierowane ze sterownika do jednostki dyskowej, są zwykle buforowane bramkami 7438. Odbiornikami sygnałów sterujących są bramki z wejściem typu przerzutnik Schmitta (np. 74LS14) bądź bramki trójstanowe (np. 74LS244). Impulsy zegarowe o wymaganych częstotliwościach (2 MHz, 1 MHz i 0,5 MHz) uzyskuje się z generatora z rezonatorem kwarcowym 8 MHz i dzielnika częstotliwości. Opis złącza umożliwiającego połączenie sterownika z jednostką dyskową przedstawiono na rys. 3.32.



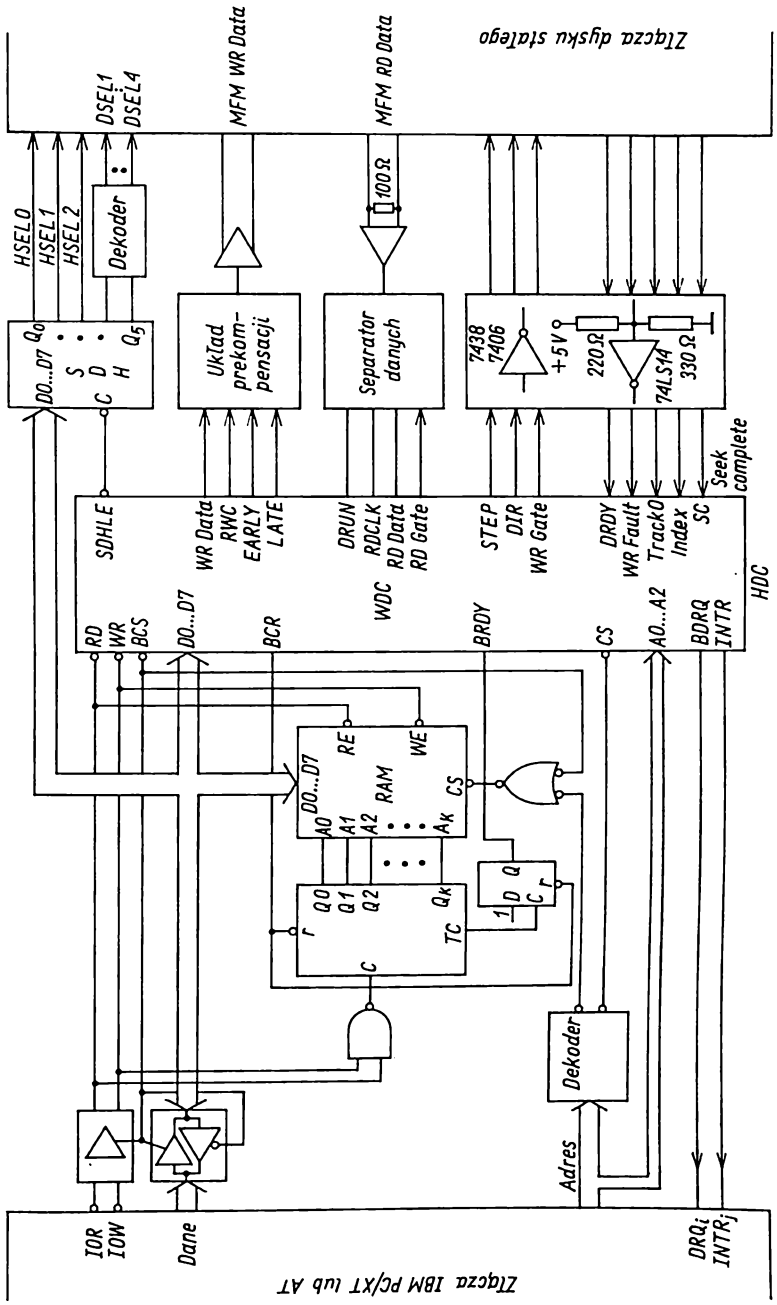
Rys. 3.32. Złącze dysku elastycznego

Do budowy *sterowników dysków stałych* stosuje się scalone sterowniki WDC (ang. *winchester disk controller*) np. WD1010, WD2010 (firmy Western Digital) lub 82062 (firmy Intel). Umożliwiają one dołączenie do czterech jednostek pamięci dyskowych typu Winchester, zawierających interfejs ST506 lub ST412. Przy zapisie stosuje się modyfikowaną metodę modulacji częstotliwości (MFM lub M^2FM). Przesłania realizuje się zwykle na zasadzie bezpośredniego dostępu do pamięci. Szybkość przesyłania danych między jednostką dyskową a sterownikiem WDC wynosi 5 Mb/s. Z tego powodu dane między sterownikiem WDC a komputerem są przesyłane przez pamięć buforową. Schemat blokowy sterownika twardych dysków przedstawiono na rys. 3.33.

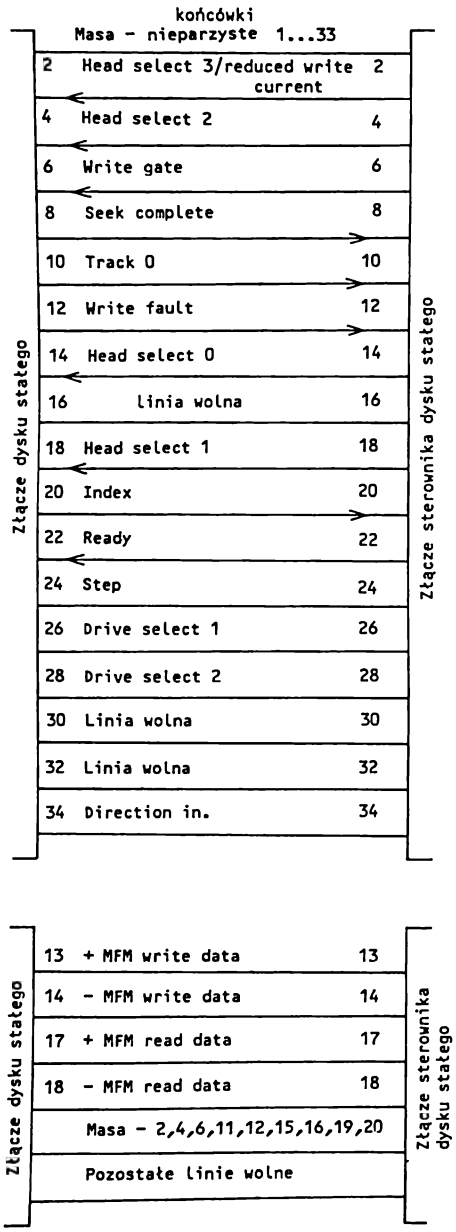
Wyróżnić w nim można sterownik WDC dysków stałych, układ sprzężenia WDC z mikrokomputerem oraz układ sprzężenia WDC z jednostką dyskową. Układ sprzężenia sterownika z mikrokomputerem zawiera bufory trójstanowe linii danych (dwukierunkowe) i linii sterujących (jednokierunkowe). Zapewniają one odpowiednią obciążalność wewnętrznej magistrali danych sterownika oraz jej separację od magistrali procesora podczas dostępu sterownika WDC do pamięci buforowej, gdy sygnał $\overline{BCS} = 0$. Procesor ma dostęp do pamięci buforowej lub rejestrów wewnętrznych sterownika, gdy sygnał $\overline{BCS} = 1$. Układ dekodera określa dostęp procesora do pamięci buforowej (dla operacji zapisu lub odczytu danych na lub z dysku) lub do rejestrów sterownika WDC. W niektórych sterownikach dekoder generuje ponadto sygnał wyboru zewnętrznego w stosunku do sterownika rejestru SDH (ang. *sector, drive head register*) zawierającego numer dysku, numer głowicy i rozmiar sektora.

Pamięć buforowa pośredniczy w przekazywaniu danych między sterownikiem WDC a pamięcią operacyjną mikrokomputera. Jest to pamięć FIFO. Do jej adresowania używa się licznika dwójkowego, zapewniającego sekwencyjny dostęp do jej komórek. Licznik ten jest zerowany sygnałem $\overline{BCR} = 0$ (ang. *buffer counter reset*) przed każdym wymaganym dostępem do pamięci. Zmiana zawartości licznika następuje po każdej operacji zapisu (odczytu). Po osiągnięciu przez licznik wartości, odpowiadającej pojemności jednego sektora, jest generowany sygnał gotowości BRDY = 1 (ang. *buffer ready*). Dla uproszczenia, szczegółowe rozwiązanie układu generowania tego sygnału zostało pominięte. Sygnał gotowości wskazuje, że pamięć buforowa została zapisana lub odczytana przez procesor i jest gotowa do odczytywania lub zapisywania przez sterownik WDC. Układ sprzężenia sterownika z jednostką dyskową zawiera (podobnie jak w sterowniku dysków elastycznych) układ zapisu danych z obwodem prekompensacji, układ odtwarzania (separacji) danych oraz nadajniki i odbiorniki sygnałów sterujących i szeregowych sygnałów danych.

Układy prekompensacji zapisywania i odtwarzania danych spełniają identyczne funkcje z funkcjami w sterowniku dysków elastycznych i są podobnie rozwiązane; aczkolwiek muszą być bardziej precyzyjne (stawia się im większe wymagania ze względu na znacznie większą częstotliwość przesłań). Przesunięcie zapisywanej danej zależy od stanu sygnałów EARLY i LATE. Odpowiedni dekoder, na podstawie zawartości rejestru SDH, określa sygnały wyboru napędu i głowicy. Wyjściowe sygnały sterujące ze sterownika do jednostki dyskowej są buforowane bramkami 7438 lub 7406. Rezystory dopasowujące linii sygnałów sterujących znajdują się w jednostce dyskowej. Odbiornikami sygnałów z jednostki mogą być bramki z wejściem typu przerzutnik Schmitta z dopasowaniem równoległym. Nadajnikiem



Rys. 3.33. Schemat blokowy sterownika dysków stałych



Rys. 3.34. Złącze dysku stałego

szeregowych sygnałów danych zapisywanych na dysk jest zazwyczaj układ różnicowy 26LS31, a odbiornikiem danych odczytywanych z dysku – układ różnicowy 26LS32 (z rezystancją falową Z_0 = 100Ω na wejściu). Układ generatora z rezonatorem kwarcowym 10 MHz zapewnia synchronizację pracy sterownika. Sygnał BDRQ (ang. *buffer data reguest*) jest sygnałem żądania DMA, uaktywnianym po zapisie lub odczycie, INTR jest natomiast sygnałem przerwania, generowanym po zakończeniu polecenia i zerowanym przez odczytanie rejestru stanu sterownika. Sterowniki dysków stałych zapewniają automatyczne generowanie i sprawdzanie znaków nadmiarowego kodu cyklicznego CRC (ang. *cycle redundance code*).

Moduł sterownika dysków stałych dołączono do mikrokomputera IBM PC/XT (AT) przez jedno ze złączy krawędziowych – połączenie z jednostką dyskową przedstawiono na rys. 3.34.

Aktualnie dostępne sterowniki dysków łączą funkcje obu przedstawionych sterowników, to znaczy jeden sterownik zapewnia dołączenie zarówno dysków elastycznych (zwykle 2 jednostki), jak i stałych (1 lub 2 jednostki).

Umożliwiają one zwykle współpracę z dyskami
- elastycznymi 5,25" o pojemności dyskietek 360 KB lub 1,2 MB;
- elastycznymi 3,5" o pojemności dyskietek 720 KB lub 1,44 MB;
- stałymi o różnych pojemnościach, zawierającymi np. do 1024 cylindrów i 16 głowic zapisująco–odczytujących.

3.6. Moduły grafiki

Obrazy wyświetlane na ekranie monitora mogą być tworzone w dwóch trybach
- graficznym.
- znakowym.

W *trybie graficznym* obraz jest tworzony z kilkuset linii, z których każda zawiera po kilkaset punktów (ang. *pixels, picture elements*). Liczba linii i punktów w każdej z nich określa tzw. *rozdzielczość*, przy czym rozróżnia się:
- małą rozdzielczość (100 x 160);
- średnią rozdzielczość (200 x 320);
- dużą rozdzielczość (200 x 640).

W *trybie znakowym* wyświetlane są znaki zakodowane w odpowiedniej matrycy, przy czym na jeden znak z odstępem przypada pole 8 x 8 lub 8 x 14 (9 x 14) punktów. Obraz można tworzyć w postaci 25 linii po 40 znaków (25 x 40) lub 25 linii po 80 znaków (25 x 80). W trybie znakowym dysponuje się zestawem 256 znaków (rys. 3.35). W trybie graficznym dysponuje się zestawem 128 znaków oraz możliwością zdefiniowania przez użytkownika dalszych 128 znaków.

Istnieje kilka rodzajów *modułów grafiki*, wzorowanych na dwu podstawowych, którymi są:
- moduł monochromatyczny MDA (ang. *monochrome display adapter*);
- moduł kolorowy CGA (ang. *color graphic adapter*).

Dziesiętnie →	0	16	32	48	64	80	96	112
↓ Szesnastkowo	0	1	2	3	4	5	6	7
0 0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1 1	☺	◄	!	1	A	Q	a	q
2 2	☻	↕	"	2	B	R	b	r
3 3	♥	!!	#	3	C	S	c	s
4 4	♦	¶	$	4	D	T	d	t
5 5	♣	§	%	5	E	U	e	u
6 6	♠	▬	&	6	F	V	f	v
7 7	•	↨	’	7	G	W	g	w
8 8	◘	↑	(	8	H	X	h	x
9 9	○	↓	)	9	I	Y	i	y
10 A	◙	→	*	:	J	Z	j	z
11 B	♂	←	+	;	K	[	k	{
12 C	♀	∟	,	<	L	\	l	¦
13 D	♪	↔	—	=	M	]	m	}
14 E	♫	▲	.	>	N	^	n	~
15 F	☼	▼	/	?	O	_	o	Δ

Dziesiętnie: 112 + 2 = 114; Szesnastkowo: 72H

126 ≙ 7EH

Rys. 3.35. Zestaw 256 znaków

Dziesiętnie	→	128	144	160	176	192	208	224	240
↑	Szesnastkowo	8	9	A	B	C	D	E	F
0	0	Ç	É	á	▒	└	╨	∝	≡
1	1	ü	æ	í	▒	─	╤	β	±
2	2	é	Æ	ó	▒	┬	╥	Γ	≥
3	3	â	ô	ù	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	┼	╘	Σ	⌠
5	5	à	ò	Ñ	╡	╞	╒	σ	⌡
6	6	å	û	a	╢	╟	╓	µ	÷
7	7	ç	ù	o	╖	╫	╪	τ	≈
8	8	ê	ÿ	¿	╕	╚	┘	Φ	°
9	9	ë	Ö	⌐	╣	╔	┌	Θ	•
10	A	è	Ü	¬	║	╩	█	Ω	·
11	B	ï	c	1/2	╗	╦	▄	δ	√
12	C	î	£	1/4	╝	╠	▌	∞	n
13	D	ì	¥	¡	╜	═	▐	φ	2
14	E	Ä	P_t	«	╛	╬	▀	∈	■
15	F	Å	f	»	┐	╧	α	∩	BLANK FF

Stopień wyjściowy monitorów realizuje się na ogół w dwóch typowych standardach
- COMPOSITE - VIDEO,
- RGB.

W pierwszym standardzie sygnał wyjściowy CV (ang. *composit-video*) tworzy się, mieszając sygnał wizji z sygnałami synchronizacji, identycznie jak w standardzie telewizyjnym, z tą tylko różnicą, że sygnał nie moduluje fali nośnej.

Standard RGB (ang. *Red*, *Green*, *Blue*) tworzy się z kilku sygnałów:
- określających kolor: R (czerwony), G (zielony), B (niebieski);
- intensywności I (ang. *intensity*);
- synchronizacji poziomej H SYN (ang. *horizontal*), pionowej V SYN (ang. *vertical*).

W monitorach RGB połączono dobrą jakość obrazu znaków monitora monochromatycznego z dużą rozdzielczością grafiki oraz kolorem. Kolory tworzy się przez połączenie trzech kolorów składowych oraz intensywności (tabl. 3.11). Jest to tzw. paleta 16-kolorowa - *paleta IRGB*. Można utworzyć paletę 64-kolorową, w której dla każdego koloru są dwa niezależne sygnały: jaśniejszy (małe litery r, g, b) i ciemniejszy (duże litery R, G, B) - *paleta RrGgBb*.

Tablica 3.11

Numer	I	R	G	B	Kolor
0	0	0	0	0	Czarny
1	0	0	0	1	Niebieski
2	0	0	1	0	Zielony
3	0	0	1	1	Cyjan (niebiesko-zielony)
4	0	1	0	0	Czerwony
5	0	1	0	1	Magenta (karmazynowy, purpurowy)
6	0	1	1	0	Brązowy
7	0	1	1	1	Biały zwykły (jasny szary)
8	1	0	0	0	Popielaty
9	1	0	0	1	Jasny niebieski
10	1	0	1	0	Jasny zielony
11	1	0	1	1	Jasny cyjan
12	1	1	0	0	Jasny czerwony
13	1	1	0	1	Jasny magenta
14	1	1	1	0	Żółty
15	1	1	1	1	Jasny biały

W trybie znakowym każdemu polu znakowemu na ekranie odpowiadają dwa sąsiednie bajty w pamięci Video RAM (pamięci obrazu). Pierwszy bajt zawiera kod znaku, który ma być wyświetlony. Drugi bajt określa atrybuty znaku (migotanie znaku - 1 bit, kolor tła znaku - 3 bity, kolor znaku - 4 bity najmniej znaczące).

W obrazie monochromatycznym wykorzystuje się 4 kombinacje bitów, odpowiadających znakowi i jego tłu, pozwalające uzyskać
- normalny biały na czarnym tle (bity znaku = 111, bity tła = 000);
- negatyw (bity znaku = 000, bity tła = 111);
- znaki podkreślone (bity znaku = 001, bity tła = 000);
- znaki niewidzialne (bity znaku = 000, bity tła = 000).

W trybie graficznym każdy punkt na ekranie może być reprezentowany w pamięci obrazu 1– lub 2–bitowo. Kodowanie bitów dla przykładowych palet jest następujące:

- standardowa paleta dla dwukolorowego trybu graficznego (0 – czarny, 1 – biały);
- standardowa paleta 0 dla czterokolorowego trybu graficznego (00 – czarny, który może być zmieniony na dowolny kolor, 01 – zielony, 10 – czerwony, 11 – brązowy);
- standardowa paleta 1 dla czterokolorowego trybu graficznego (00 – czarny, który może być zmieniony na dowolny kolor, 01 – cyjan, 10 – magenta, 11 – normalny biały).

W trybie tym nie ma możliwości uzyskania migotania znaków. W specjalistycznych zastosowaniach stosuje się paletę zawierającą np. 256 kolorów i o bardzo dużej rozdzielczości. Jeżeli w określonym standardzie, przy ustalonej pojemności pamięci strony, dysponuje się 16 kolorami (4 bity/punkt) przy rozdzielczości 200 x 160, to przy rozdzielczości 200 x 320 uzyskuje się tylko 4 kolory (2 bity/punkt), a przy rozdzielczości 200 x 640 – 1 kolor (1 bit/punkt).

W trybie graficznym poszczególne punkty są przechowywane jako ciąg bitów ze wzajemnie jednoznaczną odpowiedniością pomiędzy bitami w pamięci i punktami na ekranie.

Podstawowym układem stosowanym do realizacji modułów MDA i CGA jest popularny układ scalony CRT 6845 firmy Motorola, będący *sterownikiem Video*. Układ ten zawiera 18 programowanych rejestrów, umożliwiających sterowanie wyświetlaniem obrazu na ekranie

0H – ang. *horizontal time total*;
1H – ang. *horizontal bytes displayed*;
2H – ang. *horizontal synchronization position*;
3H – ang. *horizontal synchronization width*;
4H – ang. *vertical lines total*;
5H – ang. *vertical total adjust*;
6H – ang. *vertical lines displayed*;
7H – ang. *vertical synchronization position*;
8H – ang. *interlaced mode*;
9H – ang. *maximum character line address*;
AH – ang. *cursor start*;
BH – ang. *cursor end*;
CH – ang. *memory start address* (H);
DH – ang. *memory start address* (L);
EH – ang. *cursor* (H);
FH – ang. *cursor* (L);
10H, 11H – dla pióra świetlnego (ang. *light pen*).

W module MDA używa się czterech układów we–wy (portów), umożliwiających dostęp do czterech rejestrów w przestrzeni adresowej we–wy

3B4H – rejestr adresowy sterownika CRT 6845 (ang. 6845 *index register*);
3B5H – rejestr danych sterownika CRT 6845 (ang. 6845 *data register*);
3B8H – rejestr rodzaju pracy (ang. *display model control port*);
3BAH – rejestr stanu (ang. *display status port*).

Rejestr adresowy umożliwia zaadresowanie rejestrów sterownika CRT 6845. *Rejestru danych* używa się do przesyłania informacji do lub z rejestru, zaadresowanego przez rejestr adresowy. W *rejestrze rodzaju* pracy można ustawić 3 bity: bit 0 (1 – duża rozdzielczość), bit 3 (1 – włączyć sygnał video, 0 – wyłączyć sygnał video), bit 5 (1 – wyłączyć migotanie, 0 – włączyć migotanie). Przesłaniem wartości 2H do tego rejestru ustawia się te bity w stan normalny. *Rejestr stanu* umożliwia określenie stanu sygnału synchronizacji poziomej HSYNC (bit 0) oraz stanu kolejnych bitów sygnału video (bit 3).

Linie początku i końca kursora oraz jego położenie na ekranie określa zawartość rejestrów CRT 6845: AH, BH, EH i FH. Organizację pamięci modułu MDA przedstawiono na rys. 3.36, a jego schemat blokowy na rys. 3.37.

Moduł CGA może współpracować z monochromatycznym lub kolorowym monitorem, odbiornikiem TV oraz piórem świetlnym. W jednym z siedmiu trybów tworzenia obrazu (cztery znakowe i trzy graficzne) – tabl. 3.12 – istnieje dostęp do rejestru

3D4H – adresowego sterownika CRT 6845;
3D5H – danych sterownika CRT 6845;
3D8H – rodzaju pracy;
3D9H – wyboru koloru;
3DAH – stanu.

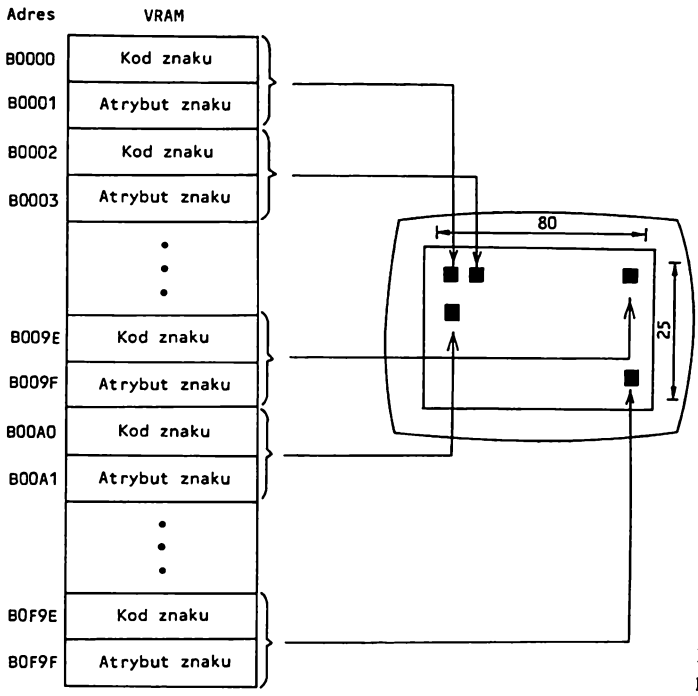
Tablica 3.12

	Tryb		Obraz
ZNAKOWY	0	25 x 40	Monochromatyczny
	1	25 x 40	Kolorowy (16 kolorów)
	2	25 x 80	Monochromatyczny
	3	25 x 80	Kolorowy (16 kolorów)
GRAFICZNY	4	200 x 320	Kolorowy (4 kolory)
	5	200 x 320	Monochromatyczny
	6	200 x 640	Monochromatyczny

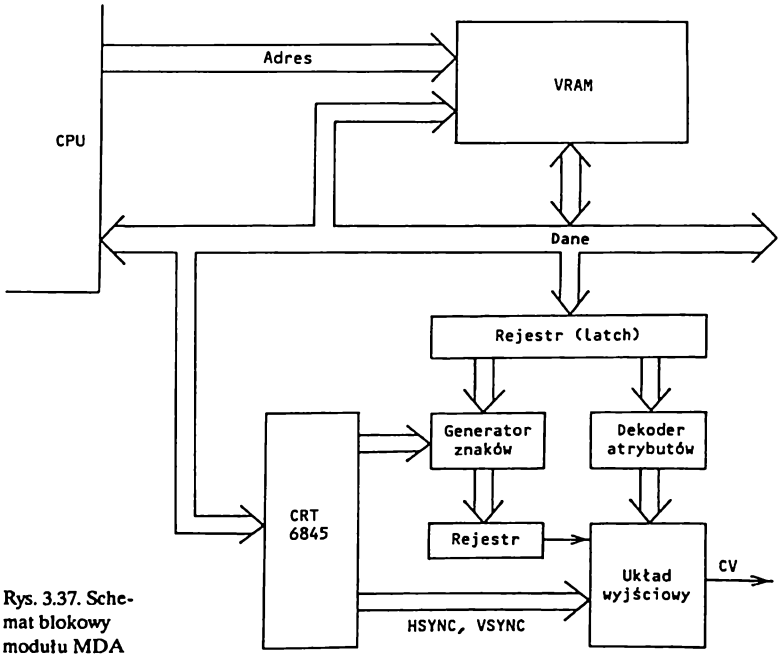
Kształt wyświetlanych znaków (8 x 8 – rys. 3.38) w trybie znakowym jest określany na podstawie informacji zapisanej w tzw. *generatorze znaków* (pamięć ROM). Pamięć obrazu zorganizowano tak, że opis znaku wyświetlanego na ekranie znajduje się w dwóch sąsiednich bajtach (organizacja dwubajtowa). Mniej znaczący bajt zawiera *kod znaku*, a bardziej znaczący – tzw. *atrybut znaku* (rys. 3.39).

Zawartości rejestrów sterownika CRT 6845 – tabl. 3.13 – określają spsób synchronizacji, wielkość i położenie obrazu oraz położenie kursora. Atrybut określa kolor znaku, kolor tła oraz sposób wyświetlania znaku.

Efektywnym i wygodnym narzędziem dla programistów jest *stronicowanie pamięci modułu CGA*; z uwagi na procedury obsługi zapewnienianc przez BIOS

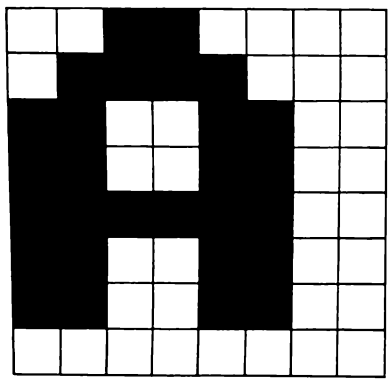


Rys. 3.36. Organizacja pamięci modułu MDA

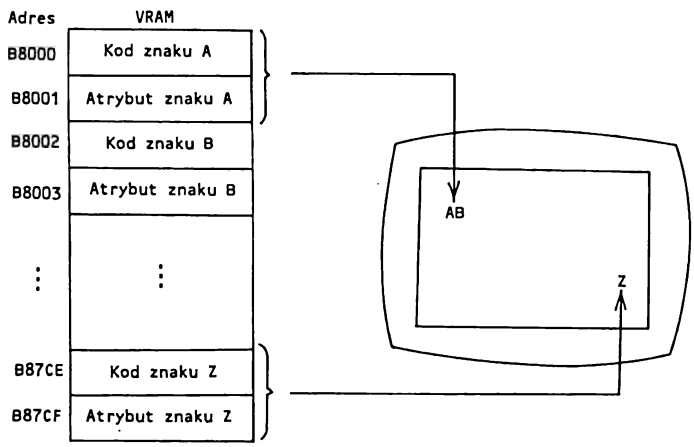


Rys. 3.37. Schemat blokowy modułu MDA

(INT 10H). Gdy obraz jest utworzony z 80 znaków w linii, wówczas są do dyspozycji 4 strony (B 8000, B 9000H, B A000H, B B000H), a gdy 40 znaków w linii – 8 stron (B 8000H, B 8800H, B 9000H, B 9800H, ..., B B000H, B B800H).



Rys. 3.38. Siatka znaku (8 x 8)



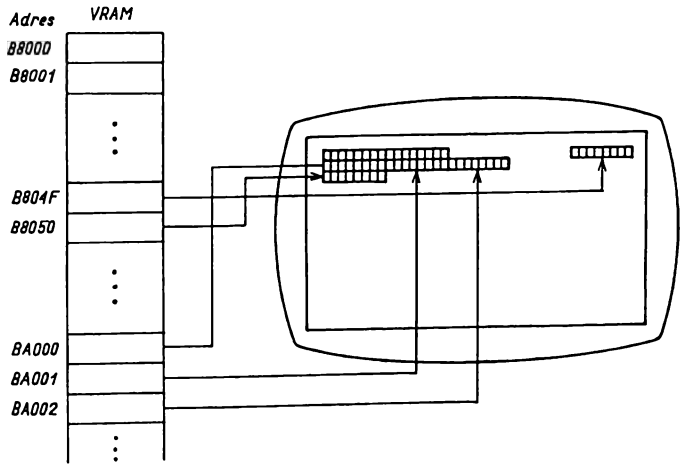
Rys. 3.39. Organizacja pamięci modułu CGA dla trybu znakowego

Adres względny 2-bajtowego pola, opisującego dany znak można wyznaczyć z zależności, określających przemieszczenie w stosunku do adresu początkowego strony.

Organizacja pamięci obrazu w trybie graficznym jest inna niż w trybie znakowym (rys. 3.40). Linie o numerach parzystych i nieparzystych są przechowywane w dwóch rozłącznych obszarach pamięci. Linia ekranu jest zapamiętywana w kolejnych 80 bajtach pamięci, co odpowiada przypisaniu każdemu punktowi 2 bitów (w trybie 200 x 320) i 1 bitu (w trybie 200 x 640) pamięci obrazu.

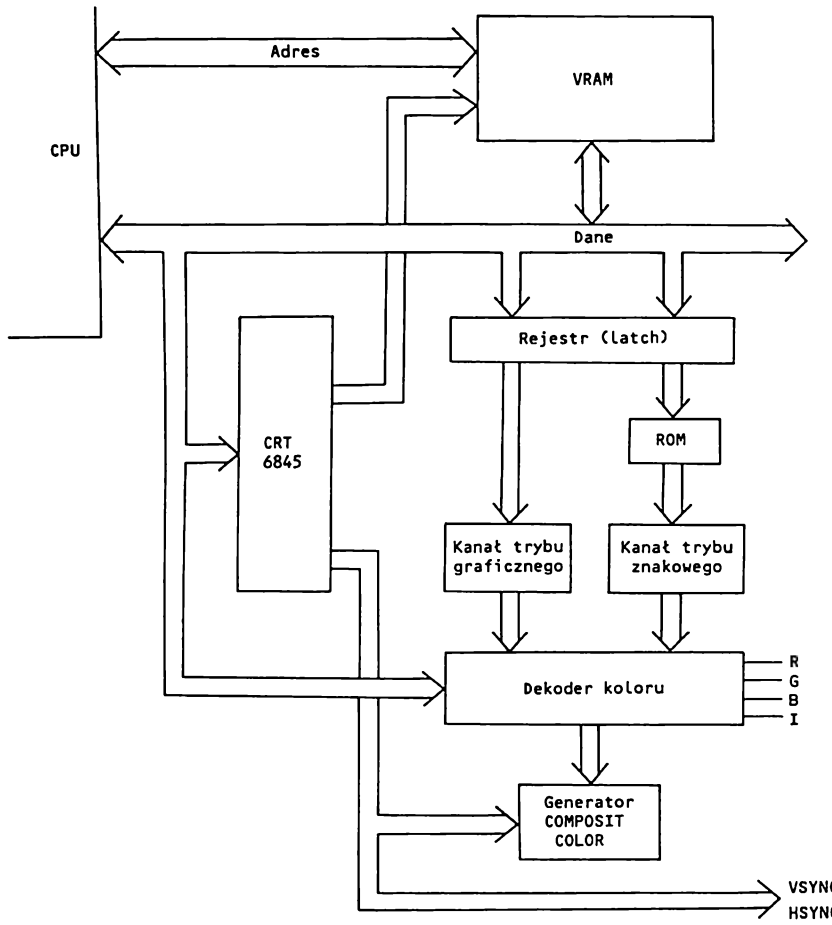
Tablica 3.13

Rejestr	Tryb		
	Graficzny	Znakowy	
		25 · 40	25 · 80
0H	38	38	71
1H	28	28	50
2H	2D	2D	5A
3H	0A	0A	0A
4H	7F	1F	1F
5H	06	06	06
6H	64	19	19
7H	70	1C	1C
8H	02	02	02
9H	01	07	07
AH	06	06	06
BH	07	07	07
CH	00	00	00
DH	00	00	00
EH	—	—	—
FH	—	—	—
10H	—	—	—
11H	—	—	—



Rys. 3.40. Organizacja pamięci modułu CGA dla trybu graficznego

Istotną cechą trybu graficznego jest możliwość wypisywania znaków na ekranie, tak jak w trybie znakowym. *Generator znaków* w pamięci RAM stwarza nieograniczone możliwości definiowania dowolnych znaków.



Rys. 3.41. Schemat blokowy modułu CGA

Tryb wyświetlania można zmieniać, ustawiając bity rejestru rodzaju pracy:

bit 0: 0 – 25 x 40, 1 – 25 x 80;
bit 1: 0 – tryb znakowy, 1 – tryb graficzny 200 x 320;
bit 2: 0 – obraz kolorowy, 1 – obraz monochromatyczny;
bit 3: 0 – wyłączyć sygnał video, 1 – włączyć sygnał video,
bit 4: 1 – tryb graficzny 200 x 640;
bit 5: 0 – wyłączyć migotanie, 1 – włączyć migotanie.

Ustawieniem bitów *rejestru wyboru koloru* można zmieniać kolor ramki w trybach znakowych oraz kolory tła i pierwszego planu w trybie graficznym

bit 0 – niebieski;
bit 1 – zielony;
bit 2 – czerwony;

bit 3 – intensywność;
bit 4 – alternatywna paleta intensywnych kolorów;
bit 5 – 0 – paleta 0, 1 – paleta 1.

Znaczenie bitów *rejestru stanu*

bit 0: (1 – dostęp do pamięci obrazu bez zakłóceń – migotania obrazu);
bit 1: (1 – ustawienie przerzutnika pióra świetlnego);
bit 2: (0 – pióro świetlne włączone, 1 – pióro świetlne wyłączone);
bit 3: (1 – powrót pionowy – można modyfikować ekran).

Schemat blokowy modułu CGA przedstawiono na rys. 3.41.

Obecnie w firmie IBM zaprzestano już produkcji modułu CGA, uznano bowiem jego parametry za niewystarczające nawet w zastosowaniach popularnych.

Znaczne rozszerzenie możliwości graficznych mikrokomputerów IBM, w stosunku do modułu CGA, uzyskano przy zastosowaniu *modułu* EGA (ang. *enhanced graphics adapter*). Umożliwia on pracę w trybie znakowym i graficznym, zapewniając także emulację modułów CGA i MDA. Wyświetlanie znaków odbywa się, w odróżnieniu do CGA, w postaci matrycy 8 x 14 punktów. Liczba kolorów jest uzależniona od wielkości pamięci obrazu. Pamięć 128 KB zapewnia pełne wykorzystanie dostępnej palety, czyli 16 kolorów wybranych dowolnie spośród 64, przy rozdzielczości 350 x 640.

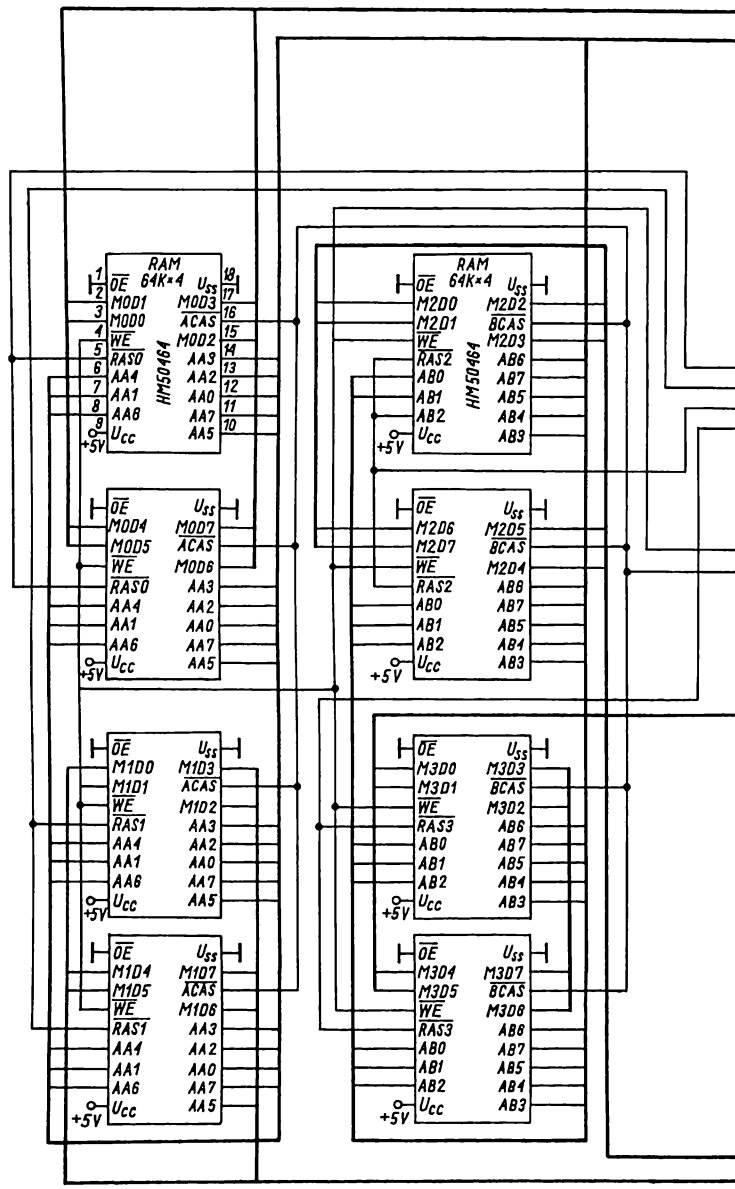
Moduł EGA zawiera cztery zasadnicze układy: sterownik CRT, sekwenter, sterownik grafiki i sterownik atrybutów – zrealizowane jako VLSI.

Sekwenter generuje sygnały sterujące pamięć obrazu i koordynuje dostęp do pamięci obrazu między jednostką centralną a sterownikiem CRT.

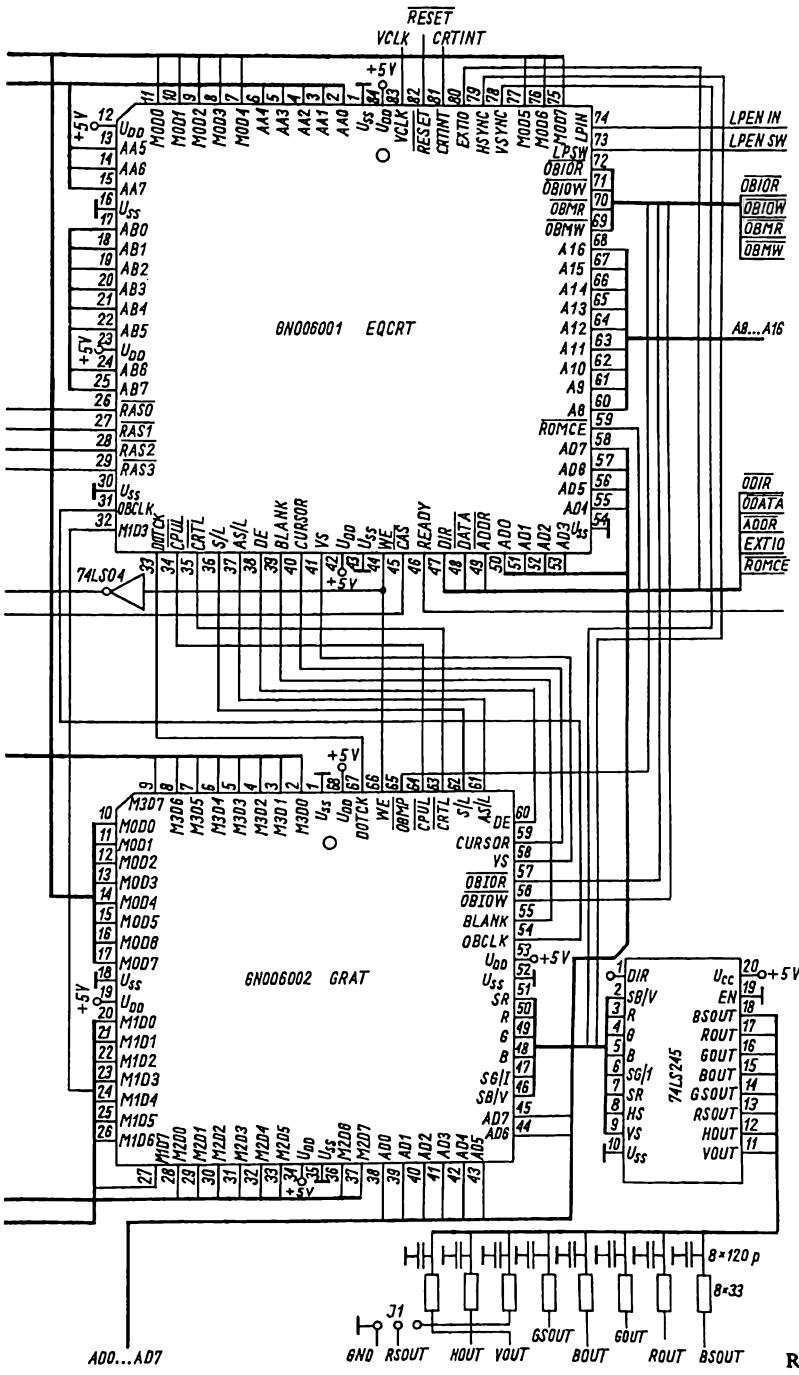
Sterownik graficzny umożliwia wykonywanie operacji na danych przy przesyłaniu danych z pamięci obrazu do sterownika atrybutów. Zawiera on 32–bitowy rejestr danych (po 1 bajcie dla każdego bloku pamięci), który zapewnia wykonanie odczytu lub zapisu 32 bitów w 1 cyklu pamięci. Każda operacja odczytu pamięci obrazu powoduje uaktualnienie jego zawartości, używanej następnie w specjalnym trybie zapisu do wykonywania 32–bitowych operacji logicznych OR, EXOR i AND z zawartością pamięci obrazu.

Tablica 3.14

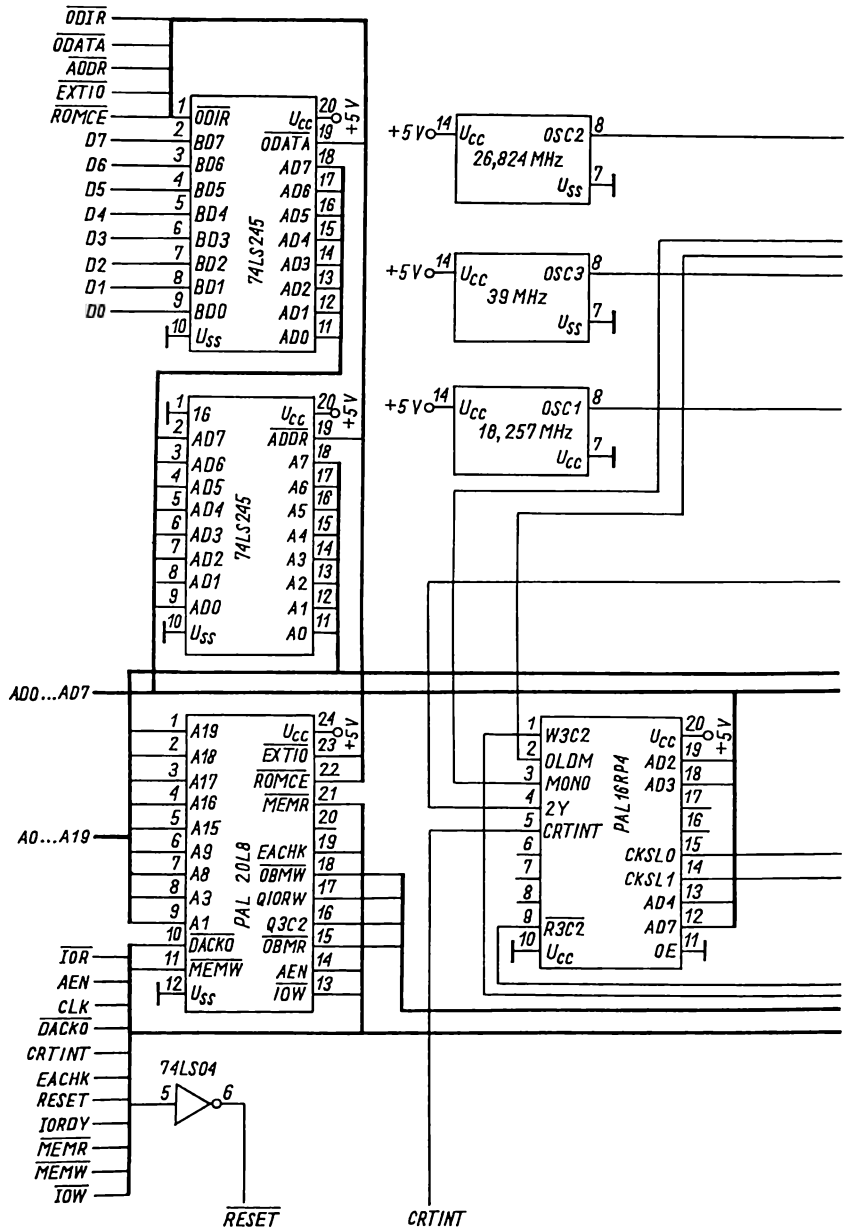
Tryb		Rozdzielczość		Wymiary	Wybór
Nr	Rodzaj	punktowa	znakowa	znaku	koloru
00/80	znakowy	350 · 320	25 · 40	8 · 14(8)	16/64(16)
01/81	znakowy	350 · 320	25 · 40	8 · 14(8)	16/64(16)
02/82	znakowy	350 · 640	25 · 80	8 · 14(8)	16/64(16)
03/83	znakowy	350 · 640	25 · 80	8 · 14(8)	16/64(16)
04/84	graficzny	200 · 320	25 · 40	8 · 8	4
05/85	graficzny	200 · 320	25 · 40	8 · 8	4
06/86	graficzny	200 · 640	25 · 80	8 · 8	2
07/87	znakowy	350 · 720	25 · 80	9 · 14	4
0D/8D	graficzny	200 · 320	25 · 40	8 · 8	16/64(16)
0E/8E	graficzny	200 · 640	25 · 80	8 · 8	16/64(16)
0F/8F	graficzny	350 · 640	25 · 80	8 · 14	4
10/90	graficzny	350 · 640	25 · 80	8 · 14	16(4)/64



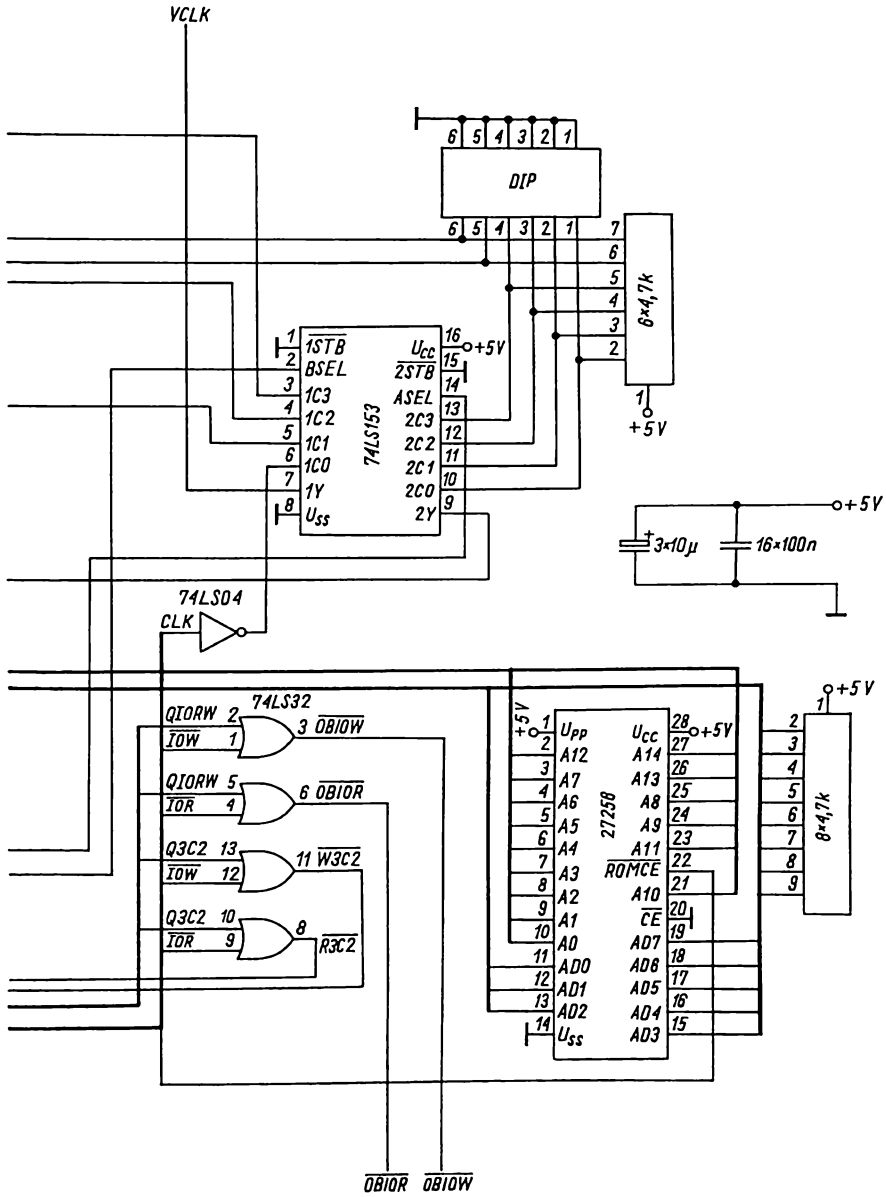
Rys. 3.42. Schemat logiczny modułu EGA


RESET
VCLK
CRTINT
+5V
GN006001 EQCRT
LPEN IN
LPEN SW
OBIOR
OBIOW
OBMR
OBMW
A8...A16
ODIR
ODATA
ADDR
EXTIO
ROMCE
74LS04
GN006002 GRAT
74LS245
8×120 p
8×33
J1
GND
RSOUT
HOUT
VOUT
GSOUT
BOUT
GOUT
ROUT
BSOUT
AD0...AD7

Rys. 3.42. cd.


ODIR
ODATA
ADDR
EXTIO
ROMCE
D7
D6
D5
D4
D3
D2
D1
D0
74LS245
+5V
26,824 MHz
39 MHz
18,257 MHz
OSC2
OSC3
OSC1
AD0...AD7
A0...A19
PAL 20L8
PAL16RP4
W3C2
OLDM
MONO
2Y
CRTINT
CKSL0
CKSL1
R3C2
OE
IOR
AEN
CLK
DACKO
CRTINT
EACHK
RESET
IORDY
MEMR
MEMW
IOW
74LS04
RESET
CRTINT

Rys. 3.42. cd.


VCLK
DIP
6 5 4 3 2 1
6 5 4 3 2 1 7
6
5
4
3
2
1
+5V
8×4,7k
1STB
BSEL
1C3
1C2
1C1
1C0
1Y
Uss
74LS153
Ucc
2STB
ASEL
2C3
2C2
2C1
2C0
2Y
+5V
3×10µ
16×100n
74LS04
CLK
74LS32
QIORW
IOW
OBIOW
IOR
OBIOR
Q3C2
W3C2
R3C2
Upp
A12
A7
A6
A5
A4
A3
A2
A1
A0
AD0
AD1
AD2
Uss
27258
Ucc
A14
A13
A8
A9
A11
ROMCE
A10
CE
AD7
AD6
AD5
AD4
AD3
8×4,7k
OBIOR
OBIOW

Rys. 3.42. cd.

Sterownik atrybutów steruje wyświetlaniem kolorów i atrybutów, zmianą rozdzielczości oraz odwzorowaniem poszczególnych kodów pamięci na ekranie. Poszczególne układy modułu EGA zawierają około siedemdziesięciu programowanych rejestrów dostępnych dla użytkownika.

Standardowe tryby pracy modułu EGA zestawiono w tabl. 3.14. Wszystkie tryby są dostępne w dwóch wersjach. W wersji z numerem mniejszym (00H...10H) kasuje się zawartość VRAM podczas przełączania, a w wersji z numerem większym (80H...90H) pozostawia się jej zawartość bez zmian.

Tryby 00H...07H odpowiadają odpowiednio trybom modułów MDA i CGA, w trybach 00H...03H emuluje się działanie analogicznych trybów modułu CGA. Jeden znak jest odwzorowany przez dwa kolejne bajty (znaku i atrybutów). Różnica polega jedynie na wyświetlaniu znaków w postaci matrycy 8 x 14 punktów z możliwością wyboru 16 dostępnych kolorów spośród 64.

Tryby 04H...06H odpowiadają trybom modułu CGA. Pamięć VRAM (dostępna, tak jak w trybach 00H...07H) odwzorowuje 2 lub 1 bit na 1 punkt ekranu (od lewej do prawej strony), przeplatając co drugi wiersz punktów między bardziej znaczącą a mniej znaczącą połową bloku pamięci 16 KB. Dzięki temu wszystkie programy, w których operuje się na pamięci obrazu modułu CGA, działają poprawnie z modułem EGA, poza tymi, w których są jawne odwołania do rejestrów sterownika CRT-6845.

Tryb 07H jest zgodny z modułem MDA. Tryby 0DH...10H stanowią rozszerzenie standardu CGA.

Pamięć obrazu podzielono na cztery bloki zwane *płatami* dla wersji modułu z pamięcią VRAM 256 KB i 128 KB oraz z pamięcią 64 KB z wyjątkiem trybów 10H i 0FH. W module z pamięcią VRAM 64 KB w trybach 10H i 0FH pamięć obrazu podzielono na dwa płaty. Numer koloru jest wówczas 2-bitowy, stąd ograniczenie do czterech kolorów.

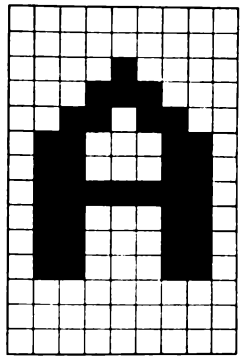
Dostęp do poszczególnych bloków pamięci przy operacjach zapisu i odczytu do lub z procesora jest przełączany przez rejestry sekwentera i sterownika graficznego.

Istnieje wiele firmowych odmian modułów EGA, wykonanych z elementów o różnym stopniu scalenia. Na rysunku 3.42 przedstawiono schemat logiczny modułu EGA, zawierającego dwa podstawowe układy scalone VLSI
- sterownik CRT z sekwenterem SEQCRT (GN 006001);
- sterownik grafiki i atrybutów GRAT (GN 006002).

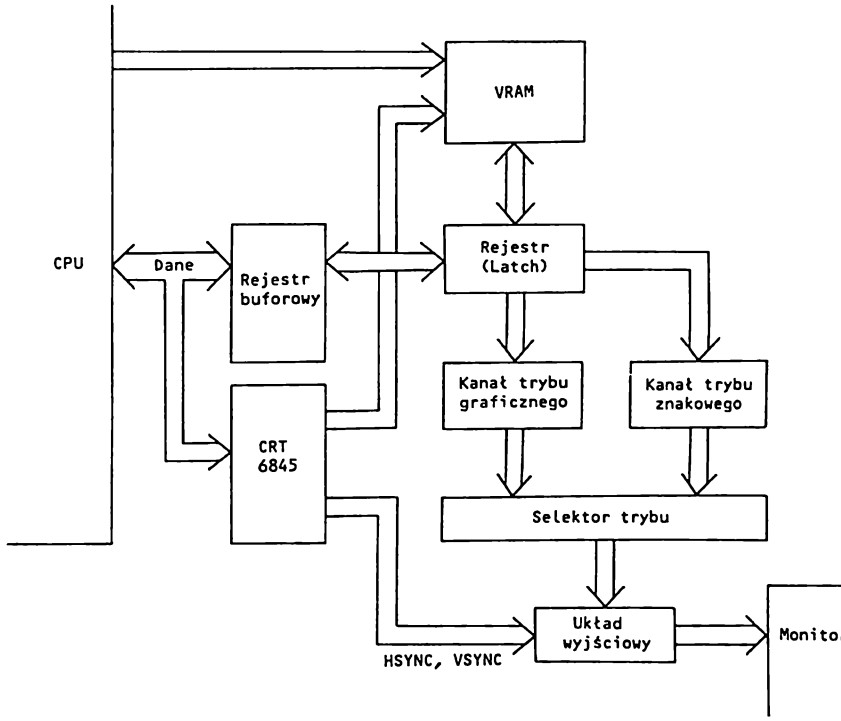
Udoskonaloną wersję modułu EGA stanowi *moduł* VGA (ang. *video graphics array*), zapewniający rozdzielczość 480 x 640, przy zachowaniu 16 kolorów.

Specyficzną odmianą modułu MDA jest *moduł* HGC (ang. *hercules graphics card*), łączący możliwości graficzne modułu CGA z lepszą jakością wyświetlanych znaków modułu MDA i dodatkowymi cechami. Ponieważ nazwa Hercules jest zastrzeżona przez firmę Computer Technology, w której moduł ten opracowano, często wykorzystuje się skrót MGA (ang. *monochrome graphics adapter*). Moduł HGC umożliwia pracę w dwóch trybach
- znakowym (25 x 80 znak w matrycy 9 x 14 : znak w polu 8 x 14 z pamięci plus odstęp; por. rys. 3.43);
- graficznym (348 x 720).

Schemat blokowy modułu HGC przedstawiono na rys. 3.44.



Rys. 3.43. Siatka znaku (9 x 14)



Rys. 3.44. Schemat blokowy modułu HGC

Istnieje wiele wersji modułów HGC, wykonanych z elementów o różnym stopniu scalenia – począwszy od ogólnie dostępnych elementów scalonych z prze-

ważającą liczbą układów MSI, a skończywszy na jednostrukturowych matrycach LSI. W trybie znakowym bufor pamięci zajmuje obszar

B0000H ... B0FFFH (strona 0);

B8000H ... B8FFFH (strona 1).

Z generatora znaków można uzyskać 256 różnych symboli. Znak może być zaciemniony (ang. *blank*), rozjaśniony (ang. *bold*), podkreślony (ang. *underline*) migotający (ang. *blink*) lub przedstawiony jako odwrotność znaku (ang. *reverse video*).

W trybie graficznym wykorzystuje się dwa przełączane bufory wizji. Zmiana zawartości pamięci obrazu, dla aktualnie wyświetlanej strony, jest natychmiast widoczna na ekranie. Bufor 64 KB podzielono na dwie strony graficzne o adresach

B0000H ... B7FFFH (strona 0)

B8000H ... BFFFFH (strona 1).

Każda ze stron zawiera jeden pełny obraz.

Wyświetlaniem obrazu steruje się poprzez, umieszczone w przestrzeni adresowej we–wy, rejestry:

3B4H – adresowy sterownika CRT 6845;

3B5H – danych sterownika CRT 6845;

3B8H – rodzaju pracy;

3BAH – stanu;

3BFH – przełączania konfiguracji (ang. *configuration switch*).

Znaczenie bitów *rejestru rodzaju pracy*

bit 1 (wybór trybu): 0 – tryb znakowy, 1 – tryb graficzny;

bit 3 (blokada wizji): 0 – zaciemnienie ekranu, 1 – włączenie wizji;

bit 5 (blokada migotania): 0 wyłączenie migotania tekstu, 1 – włączenie migotania (wpływa na każdy znak w połączeniu z atrybutem migotania);

bit 7 (wybór strony): 0 – strona 0, 1 – strona 1.

Informację o stanie ekranu prezentują określone bity *rejestru stanu*

bit 0: 0 – normalne znaki, 1 – synchronizacja pozioma, ekran ciemny;

bit 3: 0 – ekran nieaktywny, 1 – impulsy wizji;

bit 7: 0 – pionowy powrót plamki, ekran ciemny, 1 – wyświetlanie aktywne.

Mechanizm *rejestru przełączania konfiguracji* uniemożliwia przypadkową zmianę trybu pracy HGC i niewłaściwy wybór strony graficznej

bit 0: 0 – zabezpieczenie przed włączeniem trybu graficznego (maskowanie bitu 1 rejestru rodzaju pracy), 1 – zezwolenie na ustawienie trybu gra-ficznego;

bit 1: 0 – zakaz wyboru strony 1 (maskowanie bitu 7 rejestru rodzaju pracy), 1 – zezwolenie na wybranie strony 1.

Zawartość *rejestrów sterownika* CRT 6845 przedstawiono w tabl. 3.15.

Moduł HGC wyposażono również w łącze równoległe drukarki (Centronics), sterowane przez trzy rejestry we–wy o adresach

3BCH – rejestr danych (ang. *printer data port*)

3BEH – rejestr sterowania (ang. *printer control port*)

3BDH – rejestr stanu (ang. *printer status port*).

Pomocniczym elementem jest *przerzutnik pióra świetlnego* (ang. *light pen flip–flop*), ustawiany i zerowany w programie. Operacja zapisania pod adres 3B9H

w przestrzeni adresowej we–wy, powoduje ustawienie przerzutnika i zapisanie w rejestrach sterownika CRT 6845 (10H, 11H) adresu aktualnie wyświetlanego znaku. Operacja zapisania pod adres 3BBH zeruje przerzutnik. Położenie impulsu pióra świetlnego w matrycy znaku (16 x 4) w trybie graficznym określa zawartość rejestru rozszerzenia pióra świetlnego (bit 7 – numer aktywnej strony pamięci wizji; bity 6 i 5 – współrzędna pionowa impulsu o wartości od 0 do 3; bity 3...0 – współrzędna pozioma impulsu pióra świetlnego o wartości od 0 do 15.

Tablica 3.15

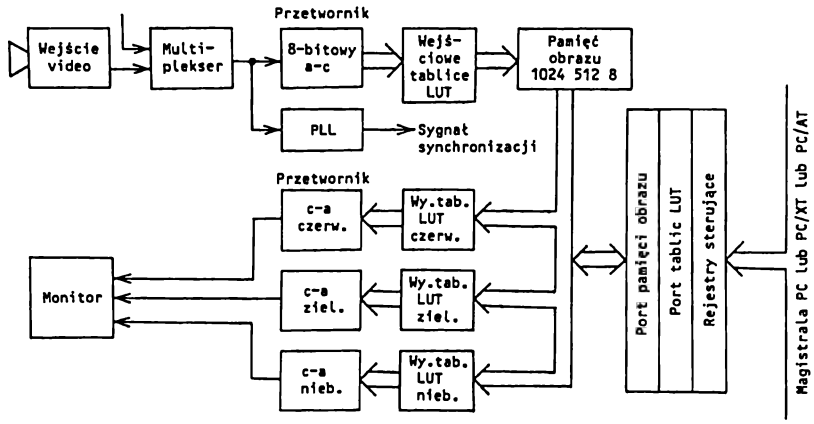
Rejestr	Tryb	
	graficzny	znakowy
0H	35H	61H
1H	3DH	50H
2H	2EH	52H
3H	07H	0FH
4H	5BH	19H
5H	02H	06H
6H, 7H	57H	19H
8H	02H	02H
9H	03H	0DH
A4	00H	0BH
BH	00H	0CH
CH...FH	00H	00H
10H...11H	—	—

Produkowane obecnie *moduły grafiki profesjonalnej* PGC (ang. *professional graphics controller*) odznaczają się najlepszą jakością grafiki obrazu.

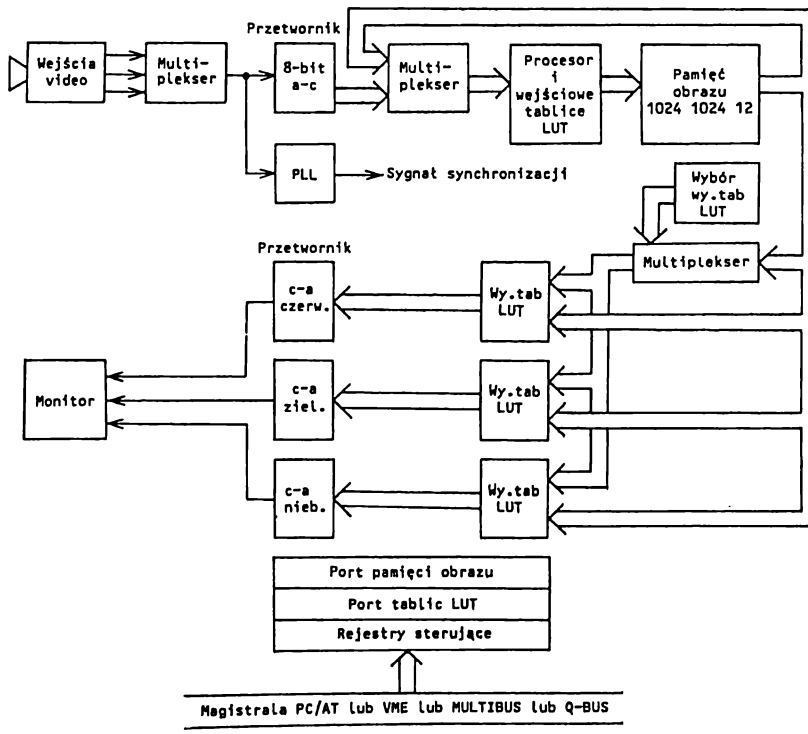
3.7. Inne moduły

Przy wykonywaniu różnorodnych zadań z użyciem mikrokomputerów IBM PC/XT lub AT, stosuje się specjalizowane moduły we–wy. Przykładem takiego modułu może być *moduł akwizycji obrazów telewizyjnych*. Moduły akwizycji obrazu produkuje się w różnych wariantach w wielu firmach na świecie. Te najprostsze to moduły jednokartowe, wkładane w złącza krawędziowe mikrokomputera. Umożliwiają one przetwarzanie analogowego obrazu z kamery telewizyjnej monochromatycznej (lub kolorowej) na postać cyfrową w układzie szybkiego przetwornika a–c (o częstotliwości przetwarzania większej bądź równej 10 MHz) i wprowadzenie go do pamięci obrazu.

Cyfrową reprezentację obrazu można przekazać do pamięci operacyjnej mikrokomputera i poddać programowej obróbce, odpowiedniej do wykonywanego zadania. Jednocześnie, pobraną z pamięci obrazu lub bezpośrednio z przetwornika a–c cyfrową reprezentację obrazu, po ponownym jej przetworzeniu na postać analogową (na sygnał wizyjny) w przetworniku c–a, można wyświetlać na monitorze z wejściem



Rys. 3.45. Moduł akwizycji obrazu telewizyjnego



Rys. 3.46. Moduł akwizycji ze wstępnym przetwarzaniem obrazu

analogowym. Przykład takiego modułu (karty) przedstawiono na rys. 3.45. Dodatkowo zawiera on wejściowe i wyjściowe tablice LUT (ang. *look up table*). Tablice te to szybkie bipolarne pamięci RAM, umożliwiające przetworzenie cyfrowych reprezentacji poszczególnych punktów obrazu (dla 8-bitowego przetwornika a–c mogą to być pamięci o organizacji 256 x 8). Zazwyczaj są one umieszczone na wejściu i wyjściu pamięci buforowej obrazu. Użycie 3 tablic LUT i 3 niezależnych przetworników c–a, na wyjściu pamięci obrazu, umożliwia wyświetlanie obrazu w pseudokolorach na monitorze kolorowym. Moduł taki zawiera zazwyczaj kilka wejść sygnału wizyjnego, dostosowanych do współpracy ze źródłem obrazu w różnych standardach (np. RS–170, CCJR).

Bardziej złożone moduły akwizycji zapewniają bardzo szybkie (w czasie rzeczywistym) wstępne przetwarzanie obrazu. Zawierają one mniej lub bardziej złożony procesor obrazu o strukturze potokowej. Przykład takiego modułu przedstawiono na rys. 3.46.

Ośmiobitowy przetwornik a–c umożliwia uzyskanie 256 stopni szarości. Moduł akwizycji obrazu kolorowego musi mieć 3 wejścia a–c dla trzech podstawowych kolorów i 3 pamięci buforowe.

4. Wybrane zagadnienia oprogramowania

Jak już wspomniano w poprzednich rozdziałach, jedną z cech charakterystycznych mikroprocesora 80286, a także nowszych mikroprocesorów 80386 i 80486, jest ich przystosowanie do pracy wielozadaniowej (ang. *multitasking*).

Ponieważ w literaturze krajowej można znaleźć wiele opracowań ([2],[10], także w serii "Mikrokomputery" [19]) na temat programowania procesorów 8086/8088 w języku asemblera oraz specyfiki (jednozadaniowego) systemu operacyjnego MS–DOS, tutaj zajmiemy się głównie procesorem 80286, a ściślej tym, czym odróżnia się on od swojego poprzednika.

4.1. Rozkazy mikroprocesora 80286

Jak dotychczas, w każdym nowym mikroprocesorze firmy Intel akceptowano zbiór rozkazów poprzednika. Lista rozkazów mikroprocesorów 80186/80188 zawiera zatem wszystkie rozkazy procesorów 8086/8088 oraz kilka bardzo przydatnych nowych rozkazów. W mikroprocesorze 80286 akceptuje się wszystkie rozkazy procesorów 80186/80188; zawarto w nim jednak nowe rozkazy, związane głównie z pracą w trybie adresowania wirtualnego.

W omówieniu rozkazów, niezbędnym do właściwego zrozumienia wielozadaniowej pracy mikroprocesora 80286, a także zamieszczonych w dalszej części przykładach programów, nawiązujemy do przedstawionego rozwoju kolejnych procesorów. Dzięki temu możemy poświęcić więcej uwagi dodatkowym, w stosunku do 8086/8088, możliwościom (i różnicom) procesora 80286 i jedynie przypomnieć listę rozkazów mikroprocesorów 8086/8088 (szczegółowo omówiona w pracy [19], nieco bardziej skrótowo zaś w [10] oraz [2]).

4.1.1. Zbiór rozkazów mikroprocesorów 8086/8088

W mikroprocesorach 8086 i 8088 realizuje się dokładnie ten sam zbiór rozkazów. Zawiera on zestaw rozkazów, typowy dla wcześniejszych procesorów 8–bitowych

(jak np. 8080/8085), a także wiele nowych, takich jak: mnożenie i dzielenie liczb dwójkowych ze znakiem i bez znaku oraz w kodzie BCD, operacje przesyłania i porównywania tablic (łańcuchów) o wielkości do 64 KB, testowanie bitów, programowe generowanie przerwań, instrukcje ułatwiające programowanie systemów wieloprocesorowych.

W rozkazach mikroprocesorów 8086/8088 w jednolity sposób traktuje się różne typy argumentów. Prawie każdy rozkaz może przetwarzać zarówno bajty, jak i słowa. Rejestry, pamięć bądź argumenty bezpośrednie można specyfikować w większości rozkazów w sposób identyczny. W szczególności, zmienne można dodawać, odejmować, przesuwać, porównywać itd. bezpośrednio w pamięci, bez konieczności pobierania ich wcześniej do rejestrów, co wpływa na znaczne zmniejszenie (w stosunku np. do mikroprocesorów 8080/8085) czasu pracy i długości programu. Także kompilatory języków wysokiego poziomu, w programach których wszystkie zmienne rezydują w pamięci, zapewniają skrócenie kodów wynikowych. Podobnie jak w innych językach asemblerowych, poszczególnym rozkazom maszynowym przypisano nazwy mnemoniczne, będące skróconym zapisem czynności wykonywanej w wyniku rozkazu.

Ogółem jest dostępnych około 100 mnemonicznych (symbolicznych) kodów rozkazów (np. MOV – ang. *move*, LEA – ang. *load effective address*, itp.), przy czym, poszczególnym kodom mnemonicznym przyporządkowano wiele różnych kodów dwójkowych, zależnych zazwyczaj od argumentów tych rozkazów. Na przykład istnieje aż 28 różnych rozkazów MOV: prześlij bajt z pamięci do rejestru, załaduj bezpośrednio słowo do rejestru itd. Utworzono ponad trzysta dwójkowych rozkazów mikroprocesorów 8086/8088. W zbiorze rozkazów można wyróżnić rozkazy: transmisji danych, arytmetyczne i logiczne, przetwarzające łańcuchy (tablice), sterujące skokami oraz wywołujące podprogramy, organizujące pętle, dotyczące przerwań i sterujące pracą procesora.

Ogólną przykładową budowę rozkazów mikroprocesorów 8086/8088 przedstawiono na rys. 2.6. Długość rozkazu wynosi od jednego do sześciu bajtów. Rozkaz może być bezargumentowy, jedno lub dwuargumentowy. Tylko jeden z argumentów może być pobrany z pamięci operacyjnej. Kombinacje argumentów:

Pierwszy argument	Drugi argument	Uwagi
–	–	Rozkaz bezargumentowy
Rejestr	–	–
Rejestr	Rejestr	–
Rejestr	Pamięć	–
Rejestr	Wartość bezpośrednia	–
Pamięć	–	–
Pamięć	Rejestr	–
Pamięć	Wartość bezpośrednia	–
Wartość bezpośrednia	–	–

Jeżeli dwuargumentowy rozkaz ma postać ogólną

kod_mnemoniczny argument_1, argument_2

to przetwarzanie argumentów odbywa się od prawego (*argument_2*) do lewego (*argument_1*), co oznacza, że wynik operacji znajduje się na miejscu lewego argumentu. Poniżej, w skróconej formie, zebrano rozkazy mikroprocesorów 8086/8088.

Rozkazy transmisji danych

A. Rozkazy transmisji danych między rejestrami oraz rejestrami a pamięcią operacyjną
MOV – prześlij bajt lub słowo;
PUSH – prześlij słowo na szczyt stosu;
POP – pobierz słowo ze szczytu stosu;
XCHG – zamień miejscami bajty lub słowa;
XLAT – pobierz bajt z tablicy.

B. Rozkazy związane z układami we-wy
IN – wprowadź bajt lub słowo z układu wejściowego;
OUT – wyprowadź bajt lub słowo do układu wyjściowego.

C. Rozkazy ładowania adresów
LEA – załaduj adres efektywny;
LDS – załaduj adres logiczny do wskazanego rejestru oraz DS;
LES – załaduj adres logiczny do wskazanego rejestru oraz ES.

D. Rozkazy związane z rejestrem znaczników
LAHF – prześlij bajt słowa znaczników do rejestru AH;
SAHF – prześlij rejestr AH do rejestru znaczników;
PUSHF – prześlij rejestr znaczników na szczyt stosu;
POPF – pobierz znaczniki ze szczytu stosu.

Rozkazy arytmetyczne i logiczne

A. Rozkazy arytmetyczne

– rozkazy dodawania
ADD – dodaj bajt lub słowo;
ADC – dodaj bajt lub słowo z przeniesieniem;
INC – dodaj 1 do bajtu lub słowa;
AAA – poprawka po dodawaniu w rozpakowanym kodzie BCD;
DAA – poprawka dziesiętna po dodawaniu w kodzie BCD;

– rozkazy odejmowania
SUB – odejmij bajt lub słowo;
SBB – odejmij bajt lub słowo z pożyczką;
DEC – odejmij 1 od bajtu lub słowa;
NEG – zaneguj lub uzupełnij do 2 bajt lub słowo;
CMP – porównaj bajt lub słowo;
AAS – poprawka po odejmowaniu w rozpakowanym kodzie BCD;
DAS – poprawka dziesiętna po odejmowaniu w kodzie BCD;

– rozkazy mnożenia
MUL – pomnóż bajt lub słowo bez znaku;
IMUL – pomnóż bajt lub słowo ze znakiem;
AAM – poprawka po mnożeniu w rozpakowanym kodzie BCD;

– rozkazy dzielenia
DIV – podziel bajt lub słowo bez znaku;
IDIV – podziel bajt lub słowo ze znakiem;
AAD – poprawka przed dzieleniem w rozpakowanym kodzie BCD;
CBW – zamień bajt na słowo;
CWD – zamień słowo na podwójne słowo.

B. Rozkazy logiczne, działające na bitach
NOT – neguj logicznie bajt lub słowo;
AND – iloczyn logiczny bajtów lub słów;
OR – suma logiczna bajtów lub słów;
XOR – nierównoważność bajtów lub słów;
TEST – testuj (porównaj logicznie) bajty lub słowa.

C. Rozkazy przesunięć
SHL – przesuń logicznie w lewo bajt lub słowo;
SAL – przesuń arytmetycznie w lewo bajt lub słowo;
SHR – przesuń logicznie w prawo bajt lub słowo;
SAR – przesuń arytmetycznie w prawo bajt lub słowo;
ROL – przesuń cyklicznie w lewo bajt lub słowo;
ROR – przesuń cyklicznie w prawo bajt lub słowo;
RCL – przesuń cyklicznie z przeniesieniem w lewo bajt lub słowo;
RCR – przesuń cyklicznie z przeniesieniem w prawo bajt lub słowo.

Rozkazy przetwarzające łańcuchy

Mikroprocesory 8086/8088 mają rozkazy, którymi można w prosty sposób operować na łańcuchach, tablicach lub blokach danych w pamięci operacyjnej. Można je podzielić na dwie grupy
– proste rozkazy realizujące pojedynczą operację;
– przedrostki deklarujące powtórzenie rozkazu prostego.

W rozkazach tych do adresowania argumentów oraz zliczania powtórzeń (kroków) używa się zawsze tych samych rejestrów
DS:SI – adres źródła danych;
ES:DI – adres wyniku (przeznaczenia);
CX – licznik powtórzeń.

Wykonanie rozkazu z przedrostkiem (rozkaz wykonywany wielokrotnie) będzie zakończone po wyzerowaniu licznika CX lub po spełnieniu innych warunków, związanych z ustawianiem znacznika zera ZF. W zależności od stanu znacznika kierunku DF, po jednokrotnym wykonaniu rozkazu, zwiększa się zawartość rejestrów SI oraz DI (DF = 0) lub zmniejsza (DF = 1), co pozwala na automatyczne zaadreso-

wanie następnego argumentu przeznaczonego do przetwarzania. W zależności od kierunku przetwarzania rejestry SI i DI zawierają offset (lub offset + długość_argumentu – 1) odpowiednich argumentów. Zawartość rejestrów SI i DI zmniejsza się lub zwiększa o 1 przy przetwarzaniu bajtów, zmniejsza się lub zwiększa o 2 natomiast przy przetwarzaniu słów.

A. Rozkazy realizujące pojedyncze operacje
- MOVS – przepisz w pamięci element łańcucha (tablicy, bloku);
- LODS – załaduj element łańcucha (tablicy, bloku) do akumulatora;
- STOS – zapamiętaj element łańcucha (tablicy, bloku) znajdujący się w akumulatorze;
- SCAS – porównaj element łańcucha (tablicy, bloku) z zawartością akumulatora;
- CMPS – porównaj elementy dwóch łańcuchów (tablic, bloków).

B. Przedrostki deklarujące powtórzenia
- REP – powtórz operację pojedynczą, aż do wyzerowania rejestru CX;
- REPZ/REPE – powtórz operację pojedynczą, aż do wyzerowania rejestru CX lub wyzerowania znacznika zera (ZF = 0);
- REPNZ/REPNE – powtórz operację pojedynczą, aż do wyzerowania rejestru CX lub ustawienia znacznika zera (ZF = 1).

Na rysunku 4.1 przedstawiono sposób realizacji powyższych rozkazów i przedrostków.

Rozkazy sterowania skokami

Podczas wykonywania rozkazów skoku zmienia się zawartość wskaźnika rozkazów IP, co umożliwia przekazanie sterowania w różne miejsca w ramach danego segmentu programu. Mówimy wtedy o tzw. *skokach "bliskich"* (ang. *near*). W przypadku tzw. *skoków "dalekich"* (ang. *far*), wykonywanych między różnymi segmentami, dodatkowo zmienia się zawartość rejestru segmentowego CS.

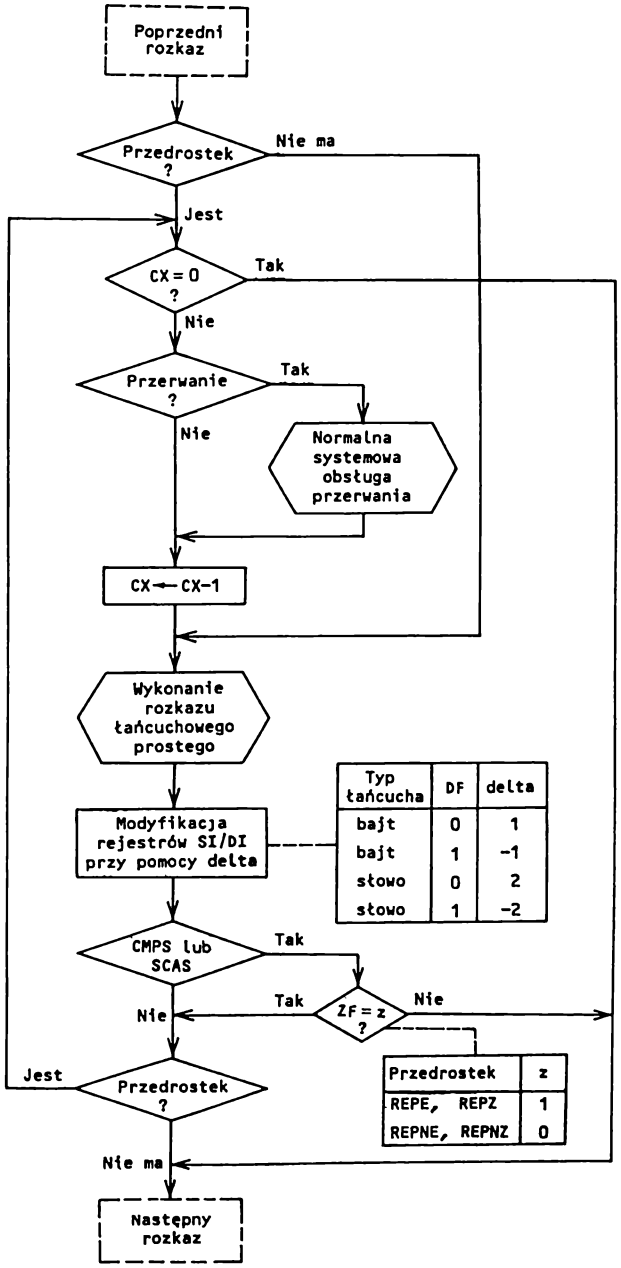
Wyróżnia się dwie grupy rozkazów skoku.

A. Rozkazy skoków bezwarunkowych
- JMP – zwykły skok bezwarunkowy;
- CALL – wywołanie podprogramu;
- RET – powrót z podprogramu do miejsca wywołania.

B. Rozkazy skoków warunkowych

– rozkazy testujące stan jednego znacznika

JC	– skok, gdy jest przeniesienie;	(CF = 1)
JNC	– skok, gdy nie ma przeniesienia;	(CF = 0)
JO	– skok, gdy jest przepełnienie;	(OF = 1)
JNO	– skok, gdy nie ma przepełnienia;	(OF = 0)
JS	– skok, gdy znak liczby jest ujemny;	(SF = 1)



Rys. 4.1. Sposób realizacji rozkazów przetwarzających łańcuchy

JNS	– skok, gdy znak liczby jest dodatni;	(SF = 0)
JP/JPE	– skok, gdy jest parzystość/parzystość parzysta;	(PF = 1)
JNP/JPO	– skok, gdy jest nieparzystość/parzystość nieparzysta;	(PF = 0)
JZ	– skok, gdy liczba równa zero;	(ZF = 1)
JNZ	– skok, gdy liczba różna od zera;	(ZF = 0)

– rozkazy związane z porównaniem dwóch argumentów

JE	– skok, gdy są równe	(ZF = 1)
JNE	– skok, gdy nie są równe, są różne	(ZF = 0)
JG	– skok, gdy jest większy	((SF⊕OF) ∨ ZF = 0)
JNG	– skok, gdy nie jest większy	((SF⊕OF) ∨ ZF = 1)
JGE	– skok, gdy jest większy lub równy	(SF⊕OF = 0)
JNGE	– skok, gdy nie jest większy lub równy	(SF⊕OF = 1)
JL	– skok, gdy jest mniejszy	(SF⊕OF = 1)
JNL	– skok, gdy nie jest mniejszy	(SF⊕OF = 0)
JLE	– skok, gdy jest mniejszy lub równy	((SF⊕OF) ∨ ZF = 1)
JNLE	– skok, gdy nie jest mniejszy lub równy	((SF⊕OF) ∨ ZF = 0)
JA	– skok, gdy jest powyżej	(CF ∨ ZF = 0)
JNA	– skok, gdy nie jest powyżej	(CF ∨ ZF = 1)
JAE	– skok, gdy jest powyżej lub równo	(CF = 0)
JNAE	– skok, gdy nie jest powyżej lub równo	(CF = 1)
JB	– skok, gdy jest poniżej	(CF = 1)
JNB	– skok, gdy nie jest poniżej	(CF = 0)
JBE	– skok, gdy jest poniżej lub równo	((CF ∨ ZF) = 1)
JNBE	– skok, gdy nie jest poniżej lub równo	((CF ∨ ZF) = 0)

Sposób wykorzystania rozkazów skoku tej grupy ilustruje następujący przykład i związana z nim tabela. Jeżeli w programie wystąpi sekwencja rozkazów

```
CMP AX, BX      ; porównaj zawartość dwóch rejestrów
Jxxx ETYKIETA   ; Jxxx – jeden z rozkazów skoków
```

to

skok wykona się, gdy	porównywane argumenty są	
	bez znaku	ze znakiem
AX = BX	JE lub JZ	JE lub JZ
AX ≠ BX	JNE lub JNZ	JNE lub JNZ
AX > BX	JA lub JNBE	JG lub JNLE
AX ⩾ BX	JAE lub JNB	JGE lub JNL
AX < BX	JB lub JNAE	JL lub JNGE
AX ⩽ BX	JBE lub JNA	JLE lub JNG

Wszystkie adresy skoków warunkowych muszą być umieszczone wewnątrz segmentu i nie mogą być odległe od adresu pierwszego bajtu następnego rozkazu o więcej niż –128 i +127. Adres skoku koduje się w postaci jednobajtowego dystansu, który w momencie wykonywania skoku jest rozszerzany do 16 bitów i dodawany (z uwzględnieniem znaku) do zawartości wskaźnika IP.

Rozkazy organizujące pętle

Rozkazy organizujące pętle wyliczane umożliwiają tworzenie konstrukcji programowych, odpowiadających konstrukcjom DO WHILE/ REPEAT UNTIL w językach wysokiego poziomu. Podczas wykonywania tych rozkazów, podobnie jak rozkazów skoków warunkowych, zmienia się wskaźnik rozkazów IP w zakresie od –128 do +127 (skoki krótkie SHORT), nie zmienia się natomiast stan rejestru wskaźników.

Licznikiem iteracji (obiegów pętli) jest rejestr CX. Do tej grupy rozkazów należą:

LOOP – powtórz sekwencję rozkazów, jeśli rejestr CX $\neq$ 0;
LOOPE/LOOPZ – powtórz sekwencję rozkazów, jeśli rejestr CX $\neq$ 0 lub jest równość/wartość równa zero;
LOOPNE/LOOPNZ – powtórz sekwencję rozkazów, jeśli rejestr CX $\neq$ 0 lub jest nierówność/wartość różna od zera;
JCXZ – skok względny, gdy rejestr CX = 0.

Rozkazy przerwań

Rozkazy przerwań umożliwiają przekazanie sterowania do podprogramów obsługi przerwań w sposób analogiczny, do stosowanego w przypadku przerwań, pochodzących od urządzeń zewnętrznych. Do tej grupy zalicza się rozkazy:

INT – skok do podprogramu obsługi przerwania;
INTO – skok do podprogramu obsługi przerwania, gdy wystąpiło przepełnienie;
IRET – powrót z podprogramu obsługi przerwania do miejsca jego wywołania.

Rozkazy sterujące procesorem

Rozkazy te umożliwiają sterowanie w programie dodatkowymi funkcjami procesora.

A. Rozkazy sterowania znaczników CF, DF i IF
STC – ustaw znacznik przeniesienia;
CLC – zeruj znacznik przeniesienia;
CMC – zaneguj znacznik przeniesienia;
STD – ustaw znacznik kierunku;
CLD – zeruj znacznik kierunku;
STI – ustaw znacznik przerwania;
CLI – zeruj znacznik przerwania.

B. Rozkazy synchronizacji zewnętrznych
HLT – zatrzymaj procesor, aż do wystąpienia przerwania;
WAIT – utrzymuj stan oczekiwania procesora, dopóki wyprowadzenie TEST będzie w stanie aktywnym;
ESC – przełącz sterowanie do zewnętrznego procesora;
LOCK – zarezerwuj dostęp do magistrali na czas wykonywania następnego rozkazu.

C. Rozkaz "nic nie rób"
NOP – nic nie rób.

4.1.2. Rozszerzenia procesora 80286

W trybie adresowania rzeczywistego w mikroprocesorze 80286 są akceptowane prawie wszystkie programy przygotowane dla mikroprocesorów 8086/8088 oraz 80186. Inny jest oczywiście czas ich wykonania, o czym wspominano już wielokrotnie. Programista powinien jednak zwrócić uwagę na niektóre problemy, z których najistotniejsze to:

- długość rozkazu wraz z przedrostkami jest ograniczona do 10 bajtów (w procesorach 8086/8088 i 80186 praktycznie nieograniczona); wielokrotne używanie przedrostków może powodować błędy;
- użycie w procesorach 8088/8086 niezdefiniowanych rozkazów, takich jak POP CS lub MOV CS, *argument* powoduje w procesorze 80286 przerwanie na szóstym poziomie;
- rozkazy przesunięć o przesunięciu większym niż 31 bitów są niedopuszczalne;
- poszczególne rozkazy wykonuje się przy mniejszej liczbie taktów zegarowych – programy, w których wykorzystuje się takty zegarowe do generowania określonych zależności czasowych, muszą być zmodyfikowane;
- przedrostek LOCK można umieszczać wyłącznie przed następującymi rozkazami: XCHG, MOV, MOVS, INS oraz OUTS;
- w procesorze 80286 wykorzystuje się 6 dodatkowych wektorów przerwań. Jeżeli wektory te były wykorzystywane w programach użytkownika, to niezbędne jest przekonstruowanie systemu;
- systemy, w których wykorzystuje się adresy układów we–wy F8H...FFH muszą być przekonstruowane. Adresy te są bowiem zarezerwowane dla koprocesora 80287 oraz dla dalszych prac rozwojowych;
- wystąpienie błędu, zidentyfikowanego w koprocesorze 80287 współpracującym z procesorem 80286, powoduje wygenerowanie przerwania na poziomie szesnastym;
- po wyzerowaniu procesora 80286, rejestry CS:IP przyjmują wartość F000:FFF0 (w 8086/8088 – FFFF:0000);
- rozkaz PUSH SP jest wykonywany inaczej w procesorze 80286. Rozkazowi PUSH SP w procesorach 8086/8088 odpowiada następująca sekwencja rozkazów wykonywana w procesorze 80286:

```
PUSH   BP
MOV    BP,SP
XCHG   BP,[BP]
```

- w przypadku wystąpienia błędu przy dzieleniu, na stos jest ładowany adres wskazujący rozkaz dzielenia, w którym ten błąd wystąpił. W procesorach 8086/8088 zapamiętany stan rejestrów CS:IP wskazuje kolejny rozkaz;
- elementy scalone mają większy stopień scalenia – a zatem jest zmieniony sposób ich programowania (np. inne adresy);
- bogatsza lista rozkazów – procesor 80286 ma rozkazy szczególnie korzystne w językach wysokiego poziomu, takich jak *Pascal* czy *C*. Celowe jest zatem ponowne tłumaczenie programów przy pomocy kompilatorów, wykorzystujących te rozkazy.

Poziom (nr) przerwania	Skrót nazwy	Opis (warunek generacji przerwania)
		W procesorach 8086/8088, 80186, 80286:
0		Dzielenie przez zero dla rozkazów DIV i IDIV, (ang. *divide error*)
1		Praca krokowa, (ang. *debug exceptions*)
2		Przerwanie niemaskowalne
3		Przerwanie wykorzystywane do ustawiania punktów zatrzymania (kontrolnych) w programach uruchomieniowych, (ang. *breakpoint*)
4		Przepełnienie – przerwanie uaktywniane rozkazem INTO, gdy znacznik OF = 1, (ang. *overflow*)
		W procesorach 80186, 80286:
5		W czasie wykonywania rozkazu BOUND stwierdzono przekroczenie limitów, (ang. *bounds check*)
		W procesorze 80286:
6	UD	Niezidentyfikowany kod rozkazu lub nieprawidłowy argument. Zapamiętany adres (CS:IP) wskazuje błędny kod, (ang. *undefined opcode*)
7	NM	Brak w systemie koprocesora arytmetycznego 80287, a w programie napotkano na rozkaz wykonywany wyłącznie przez koprocesor. W takim przypadku w obsłudze przerwania może wystąpić programowa emulacja tego rozkazu, (ang. *no math unit available*)
8	DF(E)	W jednej instrukcji wystąpiły dwa błędy, (ang. *double fault*)
9	MP	Argument arytmetyczny dla koprocesora 80287 leży poza granicą segmentu. W takiej sytuacji obsługa przerwania musi zawiesić wykonywanie zadania, a procesor zainicjować rozkazem FNINT, (ang. *math unit protection fault*)
10	TS(E)	Procesor 80286 zidentyfikował niewłaściwy (nieprawidłowy) segment zadania TS. Przyczyn błędu może być wiele, np. błąd w deskryptorze TSS, nieprawidłowy poziom uprzywilejowania i inne. Każdą przyczynę określa się specjalnym kodem błędu, przesyłanym na stos, (ang. *invalid TSS*)
11	NP(E)	Błąd sygnalizuje próbę komunikacji z segmentem, którego nie ma w pamięci operacyjnej. Przerwania używa się do ładowania segmentów z dysku do pamięci RAM systemu, (ang. *segment not present*)
12	SS(E)	Błąd dotyczący segmentu stosu. Przerwanie sygnalizuje przekroczenie wielkości stosu lub próbę komunikacji z nieistniejącym segmentem stosu, (ang. *stack segment fault*)
13	GP(E)	Wystąpił ogólny błąd związany z mechanizmem obrony. W przypadku pracy w trybie adresowania rzeczywistego wskazuje on argument (słowo), którego adres efektywny = 0FFFFH, (ang. *general protection fault*)
14, 15		Zarezerwowane dla procesora 80386.
16	MF	Błąd matematyczny koprocesora 80287, (ang. *math fault*)

W przypadku pracy procesora 80286 w trybie adresowania wirtualnego, programy uruchomione w procesorach 8086/8088 muszą być ponownie przetłu-

maczone z postaci źródłowej. W wyniku wielu rozkazów (JMP, CALL, RET itp.) wykonuje się ostatecznie tę samą operację, którą wykonano w trybie adresowania rzeczywistego, jednak powoduje to konieczność przeprowadzenia dodatkowych czynności, związanych z kontrolą ich legalności z punktu widzenia mechanizmów ochrony i poziomów uprzywilejowania. Powoduje to z kolei wydłużenie czasu wykonania tych rozkazów (w sensie liczby taktów zegarowych).

Zarówno w jednym, jak i drugim trybie adresowania błędy powstałe przy realizacji niektórych rozkazów powodują generowanie przerwań, a także (czasami) umieszczenie na stosie kodu błędu. Jest to istotne zarówno dla programistów systemowych, jak i konstruujących programy użytkowe. W tym drugim przypadku niezbędna jest wiedza o sposobie obsługi tych błędów przez system operacyjny. Przerwania związane z błędami występującymi przy wykonywaniu rozkazów zebrano w tablicy (tablica na poprzedniej stronie).

Format kodu błędu (E)

bit 0 określa typ przerwania: zewnętrzne lub programowe;
bity 1,2 określają rodzaj tablicy deskryptorów: 00–GDT, 10–LDT, 01 lub 11–IDT;
bity 3...15 zawierają indeks tablicy deskryptorów.

W niektórych przypadkach na stos jest ładowany zerowy kod błędu, tzn. wszystkie bity słowa są ustawione na zero.

Opiszemy teraz nowe rozkazy procesora 80286 oraz nowe warianty rozkazów, dostępne także w poprzednich procesorach. W opisie zastosowano następujące skróty:

arg1, arg2 – argumenty rozkazu (bajty lub słowa);
arg16 – argument 16–bitowy (rejestr lub pamięć);
mem – argument w pamięci – ogólnie;
mem32 – argument 32–bitowy w pamięci;
mem48 – argument 48–bitowy w pamięci;
reg16 – rejestr 16–bitowy;
dest – argument przeznaczenia;
src – argument źródła;
immed – argument bezpośredni;
disp – przemieszczenie (względem początku segmentu);
_low – mniej znaczący bajt;
_high – bardziej znaczący bajt;
Sreg – rejestr segmentu;
ZF – znacznik zera;
CPL – bieżący poziom uprzywilejowania segmentu CS;
DPL – poziom uprzywilejowania deskryptora;
RPL – żądany poziom uprzywilejowania, załadowany do rejestru segmentu.

Sposoby przekazywania sterowania w inne miejsca programu przy pomocy rozkazów JMP, CALL, RET, IRET lub na skutek przerwania przedstawiono w tablicy (tablica na następnej stronie).

Typ skoku	Możliwe rozkazy	Deskryptory	Tablice
Wewn. segmentu	JMP, CALL, RET	–	–
Zmiana segmentu na tym samym poziomie uprzywilejowania	JMP, CALL, RET, IRET	CS	GDT, LDT
	CALL, JMP	Bramka CALL	GDT, LDT
	INT, przerwanie	Bramka INT, TRAP	IDT
Zmiana segmentu przy innym poziomie uprzywilejowania	CALL	Bramka CALL	GDT, IDT
	INT, przerwanie	Bramka INT, TRAP	IDT
	RET, IRET	CS	GDT, IDT
Zmiana zadania	CALL, JMP, IRET	Segment TSS	GDT
	CALL, JMP	Bramka zadania	GDT, LDT
	INT, przerwanie	Bramka zadania	IDT

Rozkazy przesunięć (*)[1]

W procesorze 8086 wielkość przesunięcia określa się dwojako: bezpośrednio przy przesunięciu o 1 bit lub przez zawartość rejestru CL przy przesunięciu o więcej niż 1 bit.

W mikroprocesorach 80186, 80286 bezpośrednie określenie wielkości przesunięcia jest możliwe również przy przesunięciu o większą niż 1 liczbę bitów; na przykład są możliwe rozkazy

```
ROR AX,13          SHL SI,5                RCL BP,14
ROL ZMIENNA,2      SAL ZMIENNA[BX],6       RCR [DI],7
```

ARPL *arg16,reg16* – testowanie żądanego poziomu uprzywilejowania

Jeżeli pole *RPL* pierwszego argumentu jest mniejsze niż pole *RPL* drugiego argumentu, to znacznik zera ZF przyjmuje wartość 1, a pole *RPL* dla *arg1* przyjmuje wartość pola *RPL* dla *reg16*. W przeciwnym przypadku znacznik ZF będzie wyzerowany.

01100011	mod reg r/m	disp_low	disp_high

Liczba taktów: 10 (dla reg) / 11 (dla mem)
Przykład: ARPL AX,CX ARPL SELECTOR,DX

BOUND *reg16,arg32* – kontrola przekroczenia wartości granicznych (*)

Przy programowaniu w językach wysokiego poziomu bardzo często korzysta się z tablic (np. w języku Basic deklaracja DIM A(21), w języku C short A(21)). Do sprawdzania czy nie zostały przekroczone zadeklarowane wartości graniczne indeksów tablicy, służy rozkaz BOUND. Rozkaz ten operuje na dwóch argumentach

– pierwszym argumentem jest jeden z rejestrów ogólnych, który powinen zawierać aktualną wartość kontrolowanego indeksu (i);

[1] Gwiazdką (*) oznaczono rozkazy, które występują także w procesorze 80186.

– drugim argumentem jest adres wskazujący w pamięci operacyjnej słowo, określające najmniejszą dopuszczalną wartość indeksu (i_LOW); pod kolejnym adresem znajduje się największa dopuszczalna wartość indeksu (i_HIGH).

Rozkazem BOUND porównuje się wartość "i" z "i_LOW" oraz "i_HIGH". Jeżeli i < i_LOW lub i > i_HIGH, to jest generowane przerwanie na poziomie piątym. W przeciwnym przypadku (nie nastąpiło przekroczenie wymiarów tablicy) sekwencyjnie jest realizowany kolejny rozkaz programu.

01100010 mod reg r/m disp_low disp_high

Liczba taktów: 13 gdy nie jest generowane INT 5
Przykład: BOUND DI, BASE–4

CALL – skocz do podprogramu; różnice w wykonaniu w trybie adresowania wirtualnego

Rozkaz skoku do podprogramu w obrębie jednego segmentu nie różni się od rozkazu CALL w procesorze 8086. Przy tzw. *dalekich* (długich) *skokach* do podprogramu (przy wykorzystaniu pełnego 32–bitowego adresu logicznego), związanych ze zmianą segmentu, procesor 80286 sprawdza najpierw, czy przy adresowaniu pośrednim argument nie przekroczył wartości granicznych segmentu lub czy zawartość rejestru CS nie jest równa zero. Wystąpienie tych sytuacji powoduje wygenerowanie kodu błędu GP(0). W trybie adresowania wirtualnego procesor sprawdza następnie bajt z prawem dostępu w odpowiednim deskryptorze. W zależności od stwierdzonego błędu będzie wygenerowane przerwanie z jednym z poniższych kodów

GP(0) – przy przekroczeniu ograniczeń dla adresu;
GP(selektor) – przy braku uprawnienia dostępu do segmentu;
SS(0) – przy przekroczeniu segmentu stosu;
SS(selektor) – gdy nie istnieje (nie jest dostępny w pamięci) segment stosu;
NP(selektor) – gdy segment nie istnieje (nie jest dostępny w pamięci);
TS(0) – przy naruszeniu granic segmentu zadania TS;
TS(selektor) – gdy nie są określone uprawnienia dostępu.

Skoki długie mogą się odbywać także poprzez różnego rodzaju bramki, co zwiększa liczbę sprawdzeń warunków.

CLTS – zeruj znacznik TS

Rozkaz zeruje znacznik TS w słowie stanu procesora MSW. Można go używać wyłącznie na najwyższym (zerowym) poziomie uprzywilejowania. Znacznik TS wykorzystuje się do synchronizacji pracy procesora 80286 i koprocesorów. Wskazuje on koprocesorowi, że nastąpiła wymiana zadań.

00001111 00000110

Liczba taktów: 2

ENTER *immed16,immed8* – utwórz przestrzeń w pamięci dla zmiennych lokalnych (*)

Rozkaz ENTER odgrywa istotną rolę w programach, napisanych w językach wysokiego poziomu, takich jak Pascal lub C. W językach tych często definiuje się zmienne lokalne, ważne wyłącznie w obrębie podprogramu, w którym je zdefiniowano. Dane, używane tylko chwilowo, zapamiętuje się zazwyczaj na stosie. Ponieważ w danym podprogramie jest możliwe wywoływanie dalszych podprogramów, wygodnie jest mieć licznik, określający "głębokość" wejścia w podprogramy.

W celu łatwego tworzenia przestrzeni w obszarze stosu dla zmiennych lokalnych wykorzystuje się rozkaz ENTER. Rozkazem tym rezerwuje się odpowiednią liczbę słów w stosie i ustawia zawartość rejestrów SP i BP. Wskaźnik stosu SP wskazuje szczyt stosu (z uwzględnieniem zarezerwowanego obszaru), a rejestr bazowy BP – początek zmiennych lokalnych. Aby procedury niżej zorganizowane (zagnieżdżone) miały dostęp do zmiennych zdefiniowanych w procedurze wyżej zorganizowanej, jest konieczne przechowywanie na stosie także wskaźnika tych zmiennych.

Z rozważań tych wynika konieczność operowania dwoma argumentami w rozkazie ENTER.

- argument pierwszy jest słowem o wartości bezpośredniej i określa liczbę bajtów, niezbędną do zapamiętania zmiennych lokalnych;
- drugi argument jest bajtem o wartości bezpośredniej, definiującym poziom podprogramu.

Poniżej zamieszczono formalny zapis operacji, wykonywanych przy realizacji rozkazu ENTER. Zastosowana notacja jest zgodna ze stosowaną w pracach [10], [19].

SP ← SP— 2; next M[SP + 1] ◇ M[SP] ← BP; next HF ← SP;
next {Poziom podprogramu > 0} ⇒
⇒ (({powtórz "poziom – 1" razy} BP ← BP – 2; SP ← SP – 2;
next M[SP + 1] ◇ M[SP] ← M[BP + 1] ◇ M[BP]);
next SP ← SP – 2; next M[SP + 1] ◇ M[SP] ← HF);
next BP ← HF; SP ← SP – liczba_rezerwowanych_bajtów

Ze względów technicznych w tekście książki zastosowano symbol ◇ zamiast □ – symbolu połączenia (konkatenacji).

Trzeba pamiętać, że rozkaz ENTER nie zastępuje rozkazów PUSH oraz CALL, które należy stosować zgodnie z ogólnie przyjętymi regułami.

11001000	data_low	data_high	data

Liczba taktów:	11	ENTER immed16,0
	15	ENTER immed16,1
	12 + 4(n–1)	ENTER immed16,immed8

patrz przykład dla rozkazu LEAVE

IMUL *reg16,immed*
IMUL *reg16,reg16,immed*
IMUL *reg16,mem16,immed* – pomnóż liczby całkowite (*)

Podobnie jak w rozkazach przesunięć, w rozkazie IMUL procesora 80286 można operować na argumentach bezpośrednich, podczas gdy "normalny" rozkaz IMUL koduje się tak, jak w procesorze 8086 (np. IMUL CL lub IMUL BX), a wynik przesyła

się odpowiednio do akumulatora AX lub rejestrów DX ◇ AX. Nowy wariant rozkazu ma zmienioną składnię, wymagane są bowiem aż trzy argumenty

- pierwszy argument wskazuje jeden z 16–bitowych rejestrów ogólnych, do którego wpisuje się wynik mnożenia;
- drugim argumentem jest rejestr lub komórka w pamięci operacyjnej (mnożna);
- trzecim argumentem jest wartość bezpośrednia (mnożnik); jeżeli argument ten jest bajtem, to będzie rozszerzony do słowa (z zachowaniem znaku).

W opcji rozkazu IMUL *reg16,immed* wynik jest wpisywany w miejsce pierwszego argumentu. Wykonaną operację można zatem zapisać

argument_1 ← argument_2 * argument_3

W przypadku, gdy wynik nie mieści się w 16-bitowym rejestrze przeznaczenia, ustawia się odpowiednio znaczniki OF i CF.

011010s1	mod reg r/m	disp_low	disp_high	data_low	data_high

Liczba taktów: 21 (dla reg) / 24 (dla mem)
Przykład: IMUL BX, 25 IMUL BX, [SI],18
IMUL CX, SI,123 IMUL DX, ZMIENNA[BP],2

INS [ES:]dest,DX
INSB
INSW – wprowadź łańcuch bajtów/słów (*)

Oprócz rozkazu IN w procesorze 80286 dysponuje się rozkazem INS (ang. *input string*), którym, podobnie jak innymi rozkazami operującymi na łańcuchach, zwiększa się lub zmniejsza (zależnie od znacznika kierunku DF) zawartość rejestru indeksowego DI oraz zmniejsza zawartość licznika CX. Rozkaz INS można używać łącznie z przedrostkiem repetycji REP. Rozkaz INSB pobiera bajt, a INSW – słowo z układu wejściowego, adresowanego za pośrednictwem rejestru DX, i wprowadza go do pamięci pod adres określony zawartością rejestrów ES:DI. Zawartość rejestru DI modyfikuje się o 1 lub o 2 w zależności od tego, czy wprowadzono bajt, czy słowo. Rozkaz z przedrostkiem REP INSB (REP INSW) umożliwia wprowadzenie takiej liczby bajtów (słów), jaką zadeklarowano w rejestrze (liczniku) CX.

0110110w

Liczba taktów: 5
Przykład: INS ES:INSTR, DX REP INSB

INT
INTO – skocz do podprogramu obsługi przerwania; różnice w wykonaniu w trybie adresowania wirtualnego

W trybie adresowania rzeczywistego rozkaz INT jest wykonywany tak, jak w procesorze 8086. W trybie adresowania wirtualnego przerwanie może oznaczać zmianę realizowanego zadania, wówczas procesor wykonuje wiele sprawdzeń, w wyniku których mogą być wygenerowane dalsze przerwania.

Przerwania te są związane z następującymi kodami błędów:

GP(nr_wektora *8+2+ext)	– przy naruszeniu ograniczeń w tablicy IDT, nieprawidłowym uprawnieniu dostępu lub błędnej bramce dla przerwania;
NP(nr_wektora *8+2+ext)	– gdy nie istnieje bramka dla przerwania;
GP(0)	– przy błędnej wartości w rejestrze IP;
GP(ext)	– gdy selektor jest równy zero;
GP(selektor+ext)	– przy naruszeniu ograniczeń oraz nieprawidłowych uprawnieniach w tablicach GDT lub LDT;
NP(selektor+ext)	– gdy nie istnieje selektor;
TS(SS_selektor+ext)	– przy naruszeniu ograniczeń oraz nieprawidłowych uprawnieniach;
SS(SS_selektor+ext)	– gdy nie istnieje selektor;
SS(0)	– gdy zostały przekroczone granice stosu;

przy czym: ext = 1 dla przerwań zewnętrznych (hardware), ext=0 dla przerwań wewnętrznych (software).

IRET – powróć z podprogramu obsługi przerwania; różnice w wykonaniu w trybie adresowania wirtualnego

Rozkaz IRET w trybie adresowania rzeczywistego jest realizowany tak, jak w procesorze 8086. W trybie adresowania wirtualnego przebieg wykonania rozkazu zależy od znacznika zagnieżdżania zadania NT (ang. *nested flag*). Znacznik NT wskazuje, czy określone zadanie jest powiązane z innym (poprzednim) zadaniem przez segment stanu zadania TSS. Znacznik NT jest ustawiany równocześnie z polem "selektor poprzedniego zadania" w nowym TSS i wskazuje, że pole to jest ważne (patrz rys. 4.6). Jeżeli jest to niezbędne, wykonywane są sprawdzenia, których negatywny wynik powoduje wygenerowanie przerwań z następującymi błędami:

TS(selektor)	– przy nieprawidłowym zapisie TSS;
NP(selektor)	– gdy segment nie jest obecny;
GP(0)	– przy przekroczeniu ograniczeń;
SS(0)	– przy przekroczeniu ograniczeń stosu;
GP(selektor)	– przy niewystarczającym prawie dostępu.

JMP – skocz bezwarunkowo; różnice w wykonaniu w trybie adresowania wirtualnego

Podobnie, jak w rozkazie CALL różnice w wykonaniu dotyczą rozkazu *skoku dalekiego*, tzn. *międzysegmentowego* w trybie adresowania wirtualnego. Przed wprowadzeniem nowych wartości do rejestrów CS i IP procesor wykonuje wiele sprawdzeń, których wynik może doprowadzić do wygenerowania przerwania z błędami (ich kody podano przy opisie rozkazu CALL).

LAR *reg16,arg16* – ładuj do rejestru prawo dostępu

Drugi argument rozkazu musi przedstawiać selektor. Jeżeli wskazywany deskryptor znajduje się wewnątrz odpowiedniej tablicy, a wywołujący program ma prawo dostępu,

to do bardziej znaczącego bajtu *reg16* ładuje się bajt dostępu określonego deskryptora. Mniej znaczący bajt *reg16* będzie wyzerowany, a znacznik zera ZF przyjmuje wartość jeden. W przeciwnym przypadku znacznik ZF = 0, a zawartość rejestru *reg16* nie ulega zmianie.

00001111 00000010 mod reg r/m

Liczba taktów: 14 (dla reg) / 16 (dla mem)
Przykład: LAR AX,BX LAR CX,SELEKTOR

LDS *reg16,argument*
LES *reg16,argument* – załaduj adres logiczny do rejestru oraz DS/ES; różnice w wykonaniu w trybie adresowania wirtualnego

Znaczenie rozkazu LDS odpowiada znaczeniu tego rozkazu w procesorze 8086. W trybie adresowania wirtualnego (ładuje się wszystkie 64 bity rejestru segmentowego, przy czym 16 bitów najbardziej znaczących pobiera się z *argumentu*, a pozostałe 48 z deskryptora. W związku ze zmianą segmentu, procesor wykonuje dodatkowe sprawdzenia, których wynik może doprowadzić do wygenerowania przerwania z następującymi kodami błędów:

GP(selektor)	– przy przekroczeniu ograniczeń oraz niepodanym prawie dostępu;
NP(selektor)	– gdy segment jest niedostępny w pamięci RAM;
GP(0) lub SS(0)	– gdy argument nie leży wewnątrz segmentu;
UD	– gdy argument jest rejestrem.

LEAVE – zwolnij pamięć zarezerwowaną dla zmiennych lokalnych (*)

Rozkazem LEAVE realizuje się operację odwrotną do operacji wykonanej przez rozkaz ENTER, zwalniając zarezerwowany dla zmiennych lokalnych obszar pamięci w obrębie stosu. Kopiuje się zawartość rejestru bazowego BP do wskaźnika stosu SP i odtwarza (pobierając ze stosu) starą wartość rejestru BP. Zapis formalny operacji

$$SP \leftarrow BP;\ \text{next } BP \leftarrow M[SP+1] \diamond M[SP];\ \text{next } SP \leftarrow SP+2$$

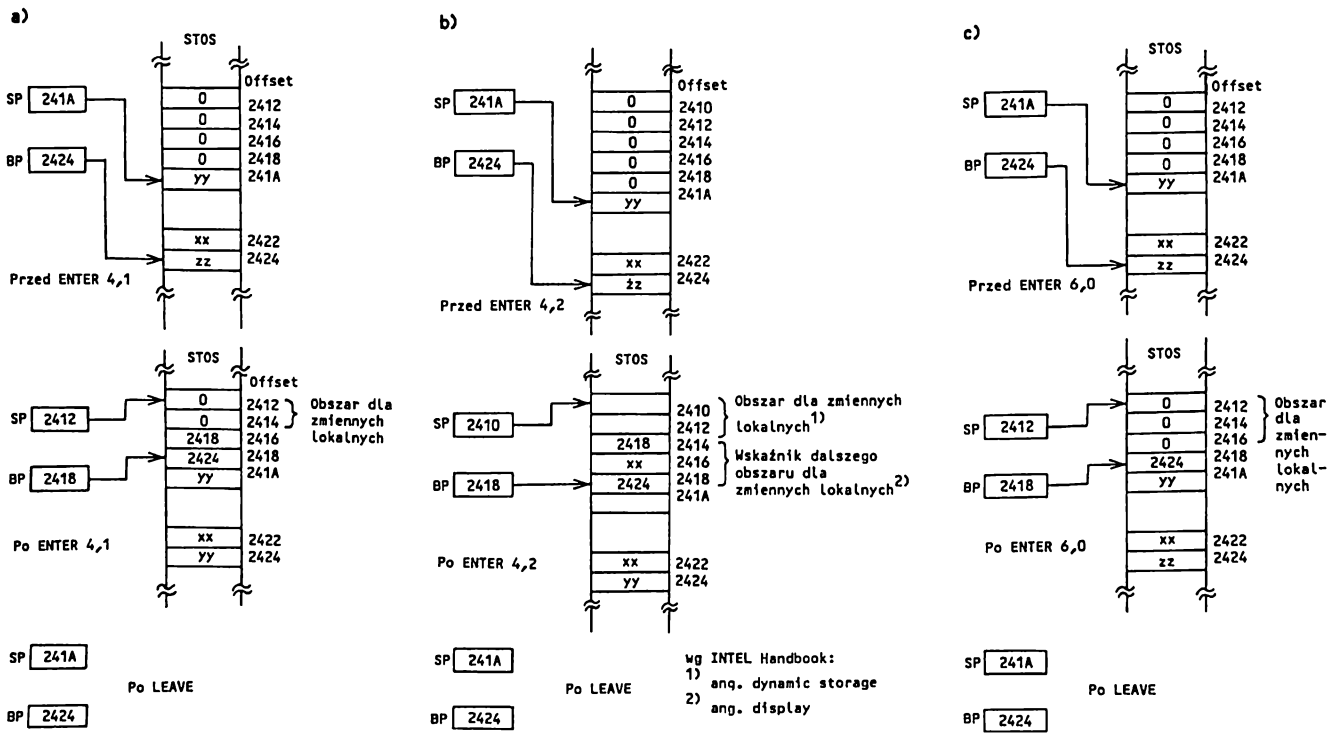
Rozkaz LEAVE nie zastępuje rozkazów POP oraz RET.

11001001

Liczba taktów: 5
Przykład: Na rysunku 4.2a, b, c przedstawiono przykładowe wykonanie rozkazów ENTER oraz LEAVE.

LGDT *mem48* – ładuj rejestr GDTR

Argumentem adresuje się 6-bajtowy blok w pamięci operacyjnej, przepisywany następnie do rejestru GDTR. Rozkaz LGDT oraz opisany niżej LIDT to jedyne rozkazy operujące rzeczywistym adresem fizycznym. Rozkazów tych nie można używać w programach użytkowych, gdyż są one zarezerwowane dla systemu operacyjnego.



Rys. 4.2. Stan rejestrów SP, BP oraz stosu przed i po wykonaniu rozkazów ENTER i LEAVE

00001111 00000001 mod 010 r/m

Liczba taktów: 11

LIDT *mem48* – ładuj rejestr IDTR

Argumentem adresuje się 6–bajtowy blok w pamięci operacyjnej, przepisywany następnie do rejestru IDTR.

00001111 00000001 mod 011 r/m

Liczba taktów: 12

LLDT *arg16*

Argumentem może być rejestr lub adres w pamięci; powinien on zawierać selektor wskazujący deskryptor w tablicy GDT. Jeżeli deskryptor ten wskazuje tablicę LDT, to odpowiednie dane deskryptora będą załadowane do rejestru LDTR.

00001111 00000000 mod 010 r/m

Liczba taktów: 17 (dla reg) / 19 (dla mem)
Przykład: LLDT AX LLDT SELECTOR

LMSW *arg16* – ładuj słowo stanu procesora

Argument rozkazu będzie załadowany do rejestru stanu procesora MSW. Rozkaz umożliwia przejście z trybu adresowania rzeczywistego do trybu adresowania wirtualnego (odwrotne przejście nie jest możliwe przy pomocy tego rozkazu).

00001111 00000001 mod 110 r/m disp_low disp_high

Liczba taktów: 3 (dla reg) / 6 (dla mem)

LSL *reg16,arg16* – ładuj rozmiar segmentu

Rozkaz powoduje załadowanie rozmiaru segmentu do rejestru *reg16*. *Arg16* musi zawierać właściwy selektor segmentu oraz wskazywać odpowiednie prawo dostępu. Jeżeli operacja była wykonana prawidłowo i *reg16* został załadowany, to znacznik ZF przyjmuje wartość 1. W przeciwnym przypadku znacznik zera będzie wyzerowany.

00001111 00000011 mod reg r/m disp_low disp_high

Liczba taktów: 14 (dla reg) / 16 (dla mem)
Przykład: LSL AX,BX LSL CX,LIMIT

LTR *arg16* – ładuj rejestr zadania TR

Rejestr zadania TR będzie załadowany zawartością podanego argumentu. Właściwy dla zadania TSS oznacza się jako aktywny. Wykonanie rozkazu nie powoduje zmiany realizowanego zadania.

00001111 00000000 mod 001 r/m disp_low disp_high

Liczba taktów: 17 (dla reg) / 19 (dla mem)
Przykład: LTR AX LTR TASK

MOV *arg1,arg2* – prześlij bajt lub słowo; różnice w wykonaniu w trybie adresowania wirtualnego

Rozkaz MOV ma znaczenie takie, jak w procesorze 8086. Jednak w przypadku, gdy *arg1* jest rejestrem segmentowym, będą załadowane wszystkie 64 bity tego rejestru (pobrane z *arg2* oraz odpowiedniego deskryptora). Konieczne jest wówczas wykonanie wielu sprawdzeń; wystąpienie błędu powoduje wygenerowanie przerwania z następującymi kodami błędów ładowanymi na stos:

GP(0)	– gdy selektor ma wartość 0 (tzn. wartość w rejestrze zawiera się między 0 i 3);
GP(selektor)	– przy przekroczeniu wartości granicznych lub niewłaściwym dostępie;
SS(selektor)	– gdy segment stosu nie jest dostępny w pamięci RAM;
NP(selektor)	– gdy segment danych (DS lub ES) nie jest dostępny w pamięci RAM.

OUTS DX,*[Sreg:]src*
OUTSB
OUTSW – wyprowadź łańcuch bajtów/słów (*)

Podobnie jak dla operacji wprowadzania z układu wejściowego, tak i dla operacji wyprowadzania do układu wyjściowego jest dostępny nowy rozkaz OUTS. Rozkaz OUTSB (OUTSW) pobiera bajt (słowo) z pamięci operacyjnej adresowanej za pośrednictwem rejestrów DS:SI i wyprowadza go do układu wyjściowego adresowanego rejestrem DX.

Analogicznie jak dla rozkazu INS, po wyprowadzeniu bajtu (słowa) zwiększa się lub zmniejsza o 1 (2) zawartość rejestru SI i zmniejsza się o 1 licznik CX. Rozkaz OUTS można używać wraz z przedrostkiem repetycji REP[1].

0110111w

Liczba taktów dla operacji pojedynczej: 5
Przykład: OUTSB
REP OUTS DX, BUFFER

PUSHA
PUSH *immed*
POPA – rozszerzone operacje na stosie

W większości współczesnych programów, pisanych w językach wysokiego poziomu, przekazywanie parametrów do podprogramów odbywa się za pośrednictwem stosu. Metodę tę zaleca się także programistom piszącym w języku asemblera. Z tego też względu rozkaz PUSH rozszerzono o nową opcję, umożliwiającą przesłanie na stos

1 Rozkazy INS oraz OUTS są wykonywane w procesorze 80286 bardzo szybko – niektóre współpracujące elementy VLSI mogą nie nadążyć i trzeba wtedy zrezygnować z rozkazów blokowych.

bezpośredniego argumentu (np. PUSH 21). Jeżeli argument jest bajtem, to będzie rozszerzony do słowa z zachowaniem znaku.

W podprogramach powinny być dostępne wszystkie rejestry procesora. W niektórych językach programowania (np. w języku C) można wręcz zadeklarować rejestry jako klasę danych, co prowadzi do znacznego przyśpieszenia wykonania programu. W celu ułatwienia i przyśpieszenia realizacji programów, w mikroprocesorze 80286 wykorzystuje się dodatkowe rozkazy PUSHA i POPA (PUSH ALL, POP ALL), umożliwiające przesłanie na stos i odczytanie ze stosu zawartości wszystkich rejestrów uniwersalnych, indeksowych i wskaźnikowych. Rozkazem PUSHA wprowadza się na stos zawartość rejestrów w podanej kolejności: AX, CX, DX, BX, SP, BP, SI i DI, przy czym w przypadku rejestru SP jest zapamiętywana wartość sprzed przesłania rejestru AX. Rozkazem POPA przywraca się poprzedni stan rejestrów, odczytując kolejnych 8 słów stosu. Odczytaną wartość rejestru SP ignoruje się.

PUSHA	01100000	
PUSH immed	011010s0	data_low data_high
POPA	01100001	

Liczba taktów:	17	PUSHA
	3	PUSH immed
	19	POPA
Przykład:	PUSH 'b'	PUSH 16239

POP *arg16* – pobierz słowo ze stosu; różnice w wykonaniu w trybie adresowania wirtualnego

Rozkaz POP ma takie samo znaczenie, jak w procesorze 8086 (wraz z rozszerzeniami procesora 80186).

Słowo odczytane ze stosu ładuje się do *arg16*, po czym wskaźnik stosu SP zwiększa się o 2. Gdy argumentem jest rejestr segmentowy, wówczas ładuje się wszystkie 64 bity. Kody błędów wykrytych w wyniku sprawdzeń określono w opisie rozkazu MOV.

RET – powróć z podprogramu; różnice w wykonaniu w trybie adresowania wirtualnego[1]

W trybie adresowania wirtualnego, gdy powrót z podprogramu wiąże się ze zmianą segmentu, procesor wykonuje wiele sprawdzeń. Ich negatywny wynik powoduje wygenerowanie przerwania z jednym z wymienionych błędów:

SS(0)	– gdy przekroczono granice stosu;
GP(0)	– przy nieprawidłowym selektorze;

---

1 W makroasemblerze *MASM*, począwszy od wersji 5.0, dopuszcza się także zapis mnemoniczny:
RETN – powróć z podprogramu *bliskiego*,
RETF – powróć z podprogramu *dalekiego*.

GP(selektor) – przy niewłaściwym prawie dostępu;
NP(selektor) – gdy segment nie jest dostępny w pamięci RAM.

SGDT *argument* – zachowaj zawartość rejestru GDTR

Zawartość rejestru GDTR będzie zapamiętana w pamięci operacyjnej (6 bajtów – najpierw rozmiar, a następnie baza), począwszy od adresu określonego *argumentem*.

00001111 00000001 mod 000 r/m

Liczba taktów: 11
Przykład: SGDT DESCRIPTOR

SIDT *argument* – zachowaj zawartość rejestru IDTR

Zawartość rejestru IDTR będzie zapamiętana w pamięci operacyjnej, (6 bajtów – najpierw rozmiar, a następnie baza) począwszy od adresu określonego *argumentem*.

00001111 00000001 mod 001 r/m

Liczba taktów: 12
Przykład: SIDT DESCRIPTOR

SLDT *arg16* – zachowaj zawartość rejestru LDTR

Zawartość rejestru LDTR (2 bajty określające indeks w tablicy GDT) będzie przesłana do podanego argumentu *arg_16*.

00001111 00000000 mod 000 r/m

Liczba taktów: 2 (dla reg) / 3 (dla mem)
Przykład: SLDT AX SLDT SELEKTOR

SMSW *arg16* – zachowaj zawartość słowa stanu procesora

Zawartość rejestru słowa stanu procesora MSW będzie przesłana do podanego argumentu *arg16*.

00001111 00000001 mod 100 r/m

Liczba taktów: 2 (dla reg) / 3 (dla mem)
Przykład: SMSW AX SMSW MACHINE

STR *arg16* – zachowaj zawartość rejestru zadania

Zawartość rejestru zadania TR będzie przesłana do podanego argumentu *arg16*.

00001111 00000000 mod 001 reg disp_low disp_high

Liczba taktów: 2 (dla reg) / 3 (dla mem)
Przykład: STR CX STR TASKREG

VERR *arg16* – sprawdź segment dla odczytu

Rozkazem tym sprawdza się, czy dany segment jest dostępny dla określonego zadania, w którym ma nastąpić odczyt (na podanym poziomie uprzywilejowania). Argument rozkazu określa selektor sprawdzanego segmentu. Po wykonaniu rozkazu znacznik zera ZF ustawia się w stan 1, gdy będą spełnione wszystkie wymienione warunki:
– selektor wskazuje na deskryptor w tablicach GDT lub LDT;
– wskazywany segment jest segmentem programu lub danych;
– segment musi być zdefiniowany jako segment do odczytu;
– poziom uprzywilejowania *DPL* musi być wyższy (lub równy) od poziomu bieżącego oraz poziomu *RPL* selektora

W przeciwnym przypadku znacznik ZF jest zerowany.

00001111	00000000	mod 100 r/m	disp_low	disp_high

Liczba taktów: 14 (dla reg) / 16 (dla mem)
Przykład: VERR AX VERR SELECTOR

VERW *arg_16* – sprawdź segment dla zapisu

Rozkazem tym kontroluje się, czy dany segment jest dostępny dla określonego zadania, w którym ma nastąpić zapis (na podanym poziomie uprzywilejowania). Argument rozkazu określa selektor sprawdzanego segmentu. Po wykonaniu rozkazu znacznik zera ZF ustawia się w stan 1, gdy spełnione są wszystkie wymienione niżej warunki:
– selektor wskazuje na deskryptor w tablicach GDT lub LDT;
– wskazywany segment jest segmentem programu lub danych;
– segment musi być zdefiniowany jako segment do zapisu;
– poziom uprzywilejowania *DPL* musi być wyższy (lub równy) niż poziom bieżący oraz poziom *RPL* selektora.

W przeciwnym przypadku znacznik ZF jest zerowany.

00001111	00000000	mod 101 r/m	disp_low	disp_high

Liczba taktów: 14 (dla reg) / 16 (dla mem)
Przykład: VERW CX VERW SELECTOR

4.2. Praca w trybie adresowania wirtualnego Przełączanie zadań

W ogólnym przypadku wykonywanie rozkazów w programie przebiega sekwencyjnie (liniowo), co oznacza pobieranie i dekodowanie kolejnych bajtów z pamięci operacyjnej. Ten sekwencyjny przebieg można przerwać rozkazami (JMP, CALL, INT) oraz odpowiednimi rozkazami powrotu z podprogramu (RET i IRET). W procesorach 8086/8088, a także w procesorze 80286, pracującym w trybie adresowania rzeczywistego, rozkazy te powodują zmianę zawartości rejestru IP, ewentualnie rejestru segmentowego CS oraz rejestru znaczników. Możliwości takie uzyskuje się także w trybie adresowania wirtualnego procesora 80286 – w którym rozkaz skoku zmienia jedynie

zawartość rejestru IP lub też oznacza przeniesienie programu do innego segmentu. Mamy wówczas do czynienia z nowym podprogramem, lecz dane i zawartości rejestrów ogólnego przeznaczenia są takie same, jak w programie, z którego nastąpił skok. W trybie adresowania wirtualnego skok między segmentami może oznaczać całkowitą zmianę wykonywanego programu (zadania). W takim przypadku wszystkie dane i rejestry *starego zadania* muszą być zapamiętane, aby po ich odtworzeniu można było kontynuować przerwany program. Dane *starego zadania* są przechowane w segmencie stanu zadania TSS, a rejestry *nowego zadania* są ładowane danymi z segmentu stanu zadania. Przy zmianie zadania jest konieczne wykonanie przez procesor wielu sprawdzeń, w wyniku których mogą być wygenerowane przerwania.

Jak już wspomniano w rozdziale drugim, procesor 80286 po wyzerowaniu go sygnałem RESET rozpoczyna pracę w trybie adresowania rzeczywistego, podobnie jak procesor 8086. Dopiero wtedy, rozkazem LMSW w programie można ustawić bit PE wprowadzający procesor 80286 w tryb adresowania wirtualnego. Podstawowe znaczenie dla tego trybu pracy ma, omówiony w rozdziale 2.3, sposób generowania adresu fizycznego z adresu wirtualnego. Jak wiadomo, adres wirtualny składa się z dwóch składników: 16–bitowego selektora, który poprzez właściwy deskryptor wybiera określony segment oraz 16–bitowego adresu efektywnego, definiującego adres względem początku segmentu. Każdy segment pamięci w systemie musi być opisany odpowiednim deskryptorem w tablicy deskryptorów. Tablicę deskryptorów (rezydującą w pamięci operacyjnej) można utworzyć następująco:

- w systemach statycznych tablice deskryptorów tworzy się, wykorzystując specjalne programy narzędziowe, np. *System–Builder* BLD286 firmy INTEL. Jest to w pewnym sensie odpowiednik programu Linker w systemach jednozadaniowych;
- w systemach dynamicznych tablice deskryptorów są tworzone oraz sterowane przez system operacyjny.

W ogólnym przypadku w systemie wielozadaniowym rezyduje wiele tablic deskryptorów.

Globalna tablica deskryptorów GDT może wystąpić w systemie tylko jeden raz i zawierać do 8192 deskryptorów, z których każdy wskazuje pewien segment w pamięci. Dopuszcza się następujące zapisy deskryptorów:

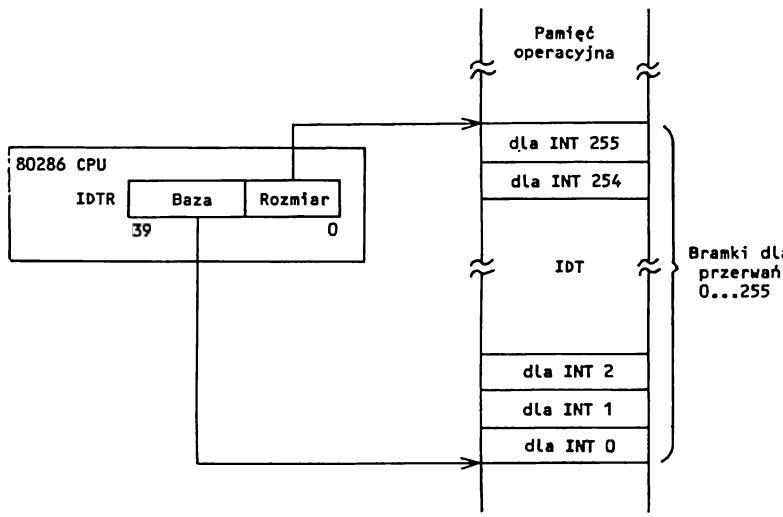
- własny deskryptor samej tablicy GDT;
- deskryptor tablicy deskryptorów przerwań IDT;
- deskryptory tablic deskryptorów lokalnych LDT;
- deskryptory segmentu stanu zadania TSS;
- deskryptory bramek;
- deskryptory globalnych segmentów danych oraz programu;
- deskryptory zerowe.

Przykładowo program Builder ładuje trzy pierwsze deskryptory w tablicy GDT w sposób standardowy (automatycznie)

deskryptor nr 0	– deskryptor zerowy;
deskryptor nr 1	– deskryptor samej tablicy GDT;
deskryptor nr 2	– deskryptor tablicy IDT;
...	– odpowiednie deskryptory, zgodnie z potrzebami systemu.

Tablica deskryptorów przerwań IDT może wystąpić w systemie także tylko jeden raz. Procesor 80286 rozróżnia 256 wektorów przerwań, przy czym każdy z nich wskazuje deskryptor w tablicy IDT (rys. 4.3). Przerwania mogą być generowane w wyniku pojawienia się zdarzenia zewnętrznego, rozkazu INT oraz błędów w wykonaniu niektórych rozkazów (np. dzielenie przez zero). Wektor przerwania określa deskryptor bramki w tablicy IDT. Deskryptor bramki wskazuje procedurę obsługi przerwania. W tablicy IDT dopuszczalne są następujące deskryptory bramek (ang. *gate descriptor*):

- przerwania (ang. *interrupt*);
- pułapki (ang. *trap*);
- zadania (ang. *task*).



Rys. 4.3. Tablica deskryptorów przerwań

W procesorze 80286 używa się przerwań o numerach 0...16 do obsługi pewnych sytuacji standardowych. Wymieniono je w poprzednim rozdziale.

Lokalna tablica deskryptorów LDT służy do logicznego rozdzielenia segmentów pamięci operacyjnej dla poszczególnych zadań. Każda tablica LDT musi mieć swój deskryptor w tablicy GDT (patrz rys. 2.43). Do określania położenia rezydujących w pamięci tablic deskryptorów służą odpowiednie rejestry. Rejestr GDTR (40–bitowy), zawierający fizyczny adres początku tablicy oraz jej wielkość, określa położenie tablicy GDT. Rejestr ten musi być załadowany odpowiednią wartością przed przełączeniem procesora w tryb adresowania wirtualnego. Podobnie, rejestr IDTR (40–bitowy) określa położenie tablicy IDT w pamięci. Rejestr LDTR (48–bitowy) określa nie tylko adres początku oraz wielkość aktualnej w danej chwili tablicy LDT, ale także reguły dostępu. Przy ładowaniu rejestru segmentowego (jako selektora) niewidoczny dla programisty blok rejestrów jest automatycznie ładowany adresowanym przez selektor deskryptorem segmentu.

W procesorze 80286 zastosowano trzy rodzaje mechanizmu ochrony poszczególnych obszarów pamięci

- izolowanie oprogramowania systemowego od programów użytkowych; zasoby systemowe są sterowane wyłącznie przez system operacyjny – możliwości adresowania pamięci operacyjnej przez użytkownika w programie są kontrolowane i ograniczone;
- izolowanie programów użytkowych między sobą; wykonywany program nie może wprowadzić błędu do innego zadania;
- kontrola typu danych – nie jest możliwe interpretowanie danych jako programu oraz modyfikowanie kodu programu.

Jak wyjaśniono to już wcześniej, segmenty są najmniejszym, spójnym (liniowym) obszarem pamięci z własnymi atrybutami (określającymi opisane wyżej zabezpieczenia). Atrybuty segmentu:

- typ segmentu (zapisany w bajcie dostępu deskryptora);
- poziom uprzywilejowania (zapisany jw.);
- rozmiar segmentu (zapisany w polu *Rozmiar* deskryptora).

Typ segmentu wynika z dyrektywy "Segment" asemblera i może mieć następującą wartość:

RO (ang. *read only*) – dane tego segmentu można tylko odczytywać;
EO (ang. *execute only*) – segment można wykonywać wyłącznie jako program;
ER (ang. *execute or read*) – segment zawiera zarówno program, jak i dane, które można tylko odczytywać;
RW (ang. *read or write*) – segment można odczytywać lub wypełniać danymi.

Podczas ładowania rejestru segmentowego jest kontrolowany typ segmentu. Na przykład, nie jest możliwe załadowanie rejestru segmentowego stosu SS selektorem, wskazującym typ segmentu RO.

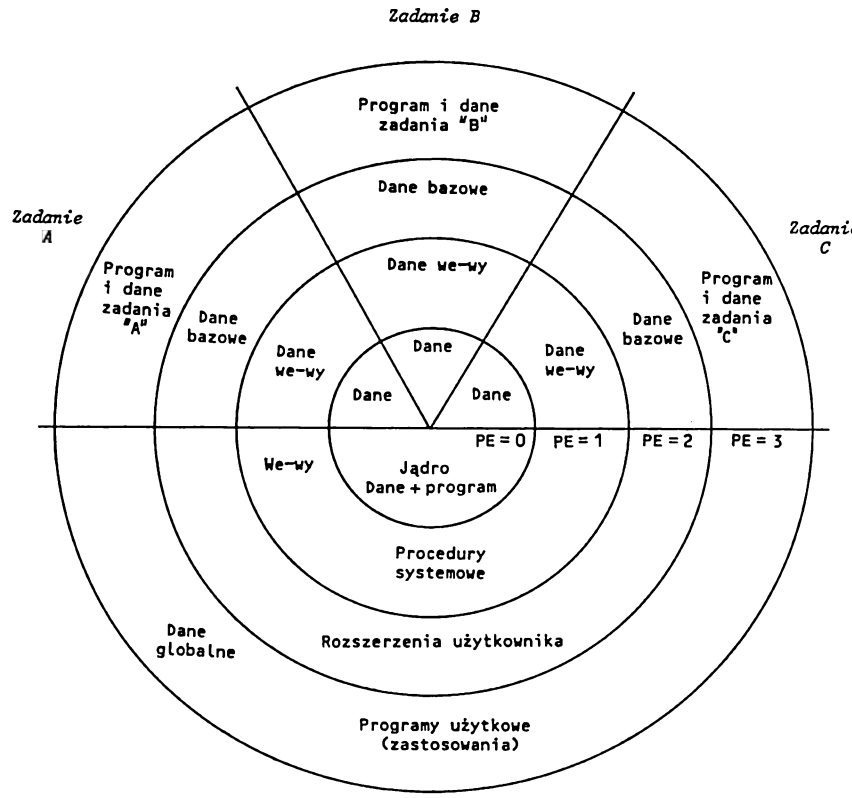
Czasami dostęp do tablic lub segmentów jest związany z kontrolą podanego w deskryptorze rozmiaru; jego przekroczenie powoduje wygenerowanie przerwania (patrz opis rozkazów procesora 80286).

Poziomy uprzywilejowania w procesorze 80286 wykorzystuje się do zabezpieczania "ważniejszych" procedur przed mniej ważnymi w obrębie jednego zadania. Przykładowo, gdyby program użytkowy miał dostęp do tablic używanych przez system operacyjny, mogło by to doprowadzić do wystąpienia błędów w pracy systemu, a nawet jego awarii.

Z tego względu segmenty jednego zadania mogą otrzymać jeden z czterech (0...3) poziomów uprzywilejowania. Najwyższy priorytet ma zerowy poziom uprzywilejowania – właściwy dla jądra systemu operacyjnego. Poziom trzeci, najniższy, przydzielono programom użytkowym. Omówione poziomy uprzywilejowania przedstawiono na rys. 4.4.

W systemie można zdefiniować wiele zadań, jednak w danym momencie procesor realizuje tylko jedno z nich. Przełączenie zadań, a więc "odłączenie" aktualnie wykonywanego zadania od procesora i przydzielenie innego, może nastąpić na skutek

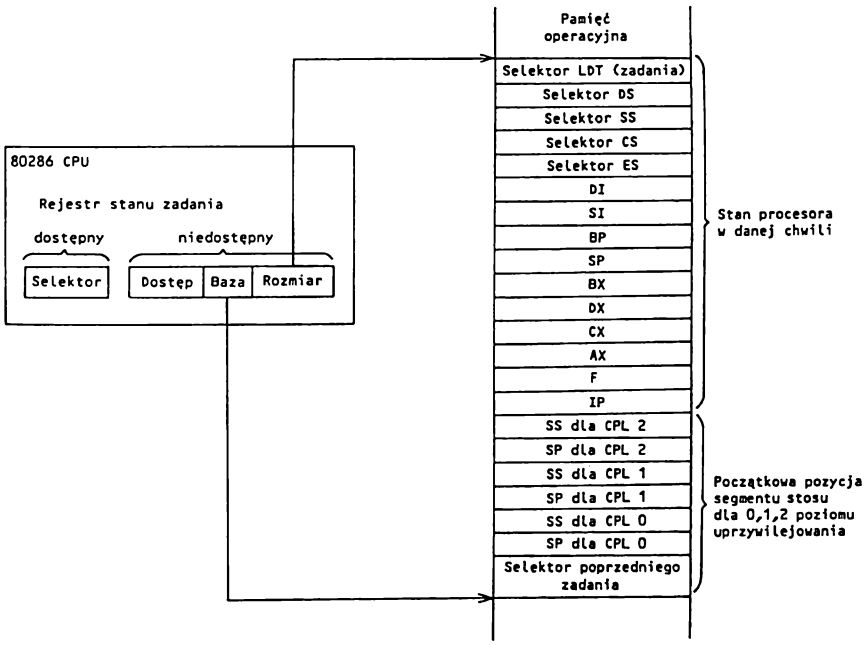
- przerwania;
- rozkazu CALL tzw. *międzyzadaniowego* (ang. *inter-task call*);
- rozkazu JMP tzw. *międzyzadaniowego* (ang. *inter-task jump*);
- rozkazu IRET.



Rys. 4.4. Poziomy uprzywilejowania w systemie

Przełączanie zadań jest najważniejszą czynnością wykonywaną w każdym systemie wielozadaniowym bądź wielodostępnym. W procesorze 80286 są dostępne specjalne szybkie układy logiczne wspomagające operację przełączania zadań, występującą stosunkowo często w systemach czasu rzeczywistego. Stan wykonywanego zadania wynika jednoznacznie z zawartości rejestrów procesora. Aby stan ten zapamiętać – co jest niezbędne, aby zadanie można było kiedyś kontynuować – każdemu zadaniu przydzielono w pamięci dodatkowy segment stanu zadania TSS. Z tym typem segmentu jest związany rejestr zadania TR, który zawiera selektor aktualnie wykonywanego zadania. Selektor ten wskazuje na deskryptor TSS w tablicy GDT. Podczas ładowania selektora do rejestru TR, deskryptor TSS jest automatycznie ładowany do niewidocznej części rejestru TR. Zawartość segmentu określonego przez ten deskryptor pokazano na rys. 4.5. Na rysunku 4.6 przedstawiono, w jaki sposób za pomocą segmentu TSS oraz tablicy LDT są opisane dwa zadania *A* i *B*. Selektor LDT, dla zadania w segmencie TSS, wskazuje na deskryptor tablicy LDT w tablicy GDT. Za pomocą tego deskryptora specyfikuje się tablicę LDT, należącą do zadania. Jeżeli teraz selektor rozkazu CALL lub JMP w zadaniu *A* użyje deskryptora segmentu TSS zadania

B, to rejestr zadania TR w procesorze będzie przeładowany, przy czym stan zadania *A* będzie zapamiętany w swoim segmencie TSS. Aktywne jest więc teraz zadanie *B*, przy czym rejestry procesora były załadowane zgodnie ze swoim segmentem TSS. Selektor segmentu poprzedniego zadania (ang. *back link selektor*) – patrz rys. 4.5 – zawiera selektor, który wskazuje deskryptor segmentu stanu zadania przerwanego (TSS). Każdy rozkaz CALL, powodujący zmianę zadania, zapisuje pole selektora poprzedniego zadania w segmencie TSS oraz ustawia znacznik NT. Dzięki temu tworzy się łańcuch połączeń między przerywanymi zadaniami, wykorzystywany przez rozkaz IRET. Rozkaz ten spowoduje przełączenie zadań jedynie wówczas, gdy bit znacznika NT jest równy 1.

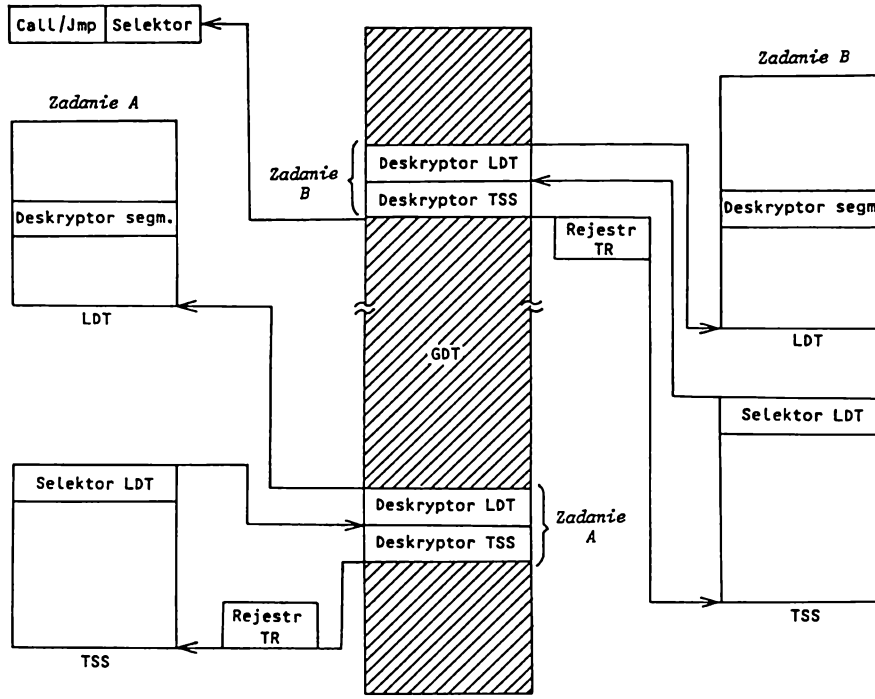


Rys. 4.5. Segment stanu zadania TSS

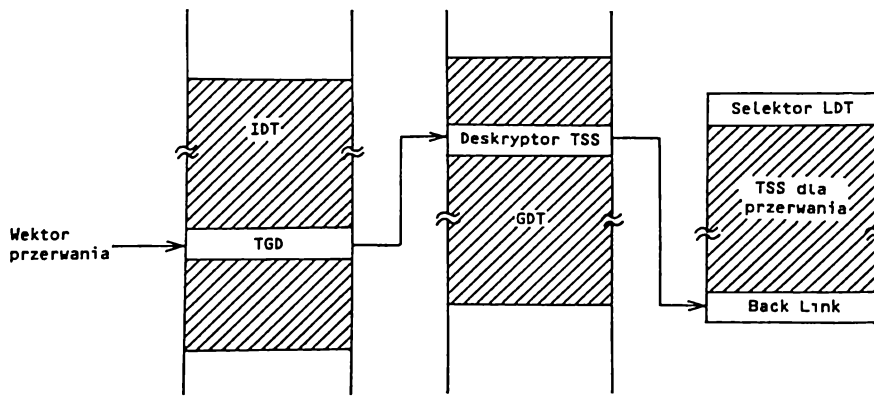
W przypadku przerwania, jeżeli wektor przerwania wskazuje na deskryptor bramki zadania, to będzie wybrany deskryptor segmentu TSS i nastąpi wymiana zadania (rys. 4.7). W przypadku bramki przerwania lub pułapki, zadanie nie zmieni się, a w procesorze będą podjęte działania, jak dla bramki rozkazu CALL (rys. 4.8).

Jest rzeczą zrozumiałą, że wymiana zadań może nastąpić jedynie wówczas, gdy wykorzystywane przy tej operacji tablice (TSS, deskryptory) są prawidłowo zapisane, a *poziomy uprzywilejowania* odpowiednie. W poprzednich rozdziałach wyróżniono trzy różne poziomy kontrolowane w czasie wymiany zadań

CPL (ang. *current privilege level*) – bieżący poziom uprzywilejowania, określony przez aktualną zawartość rejestru segmentowego CS (SS) na bitach 0 i 1;

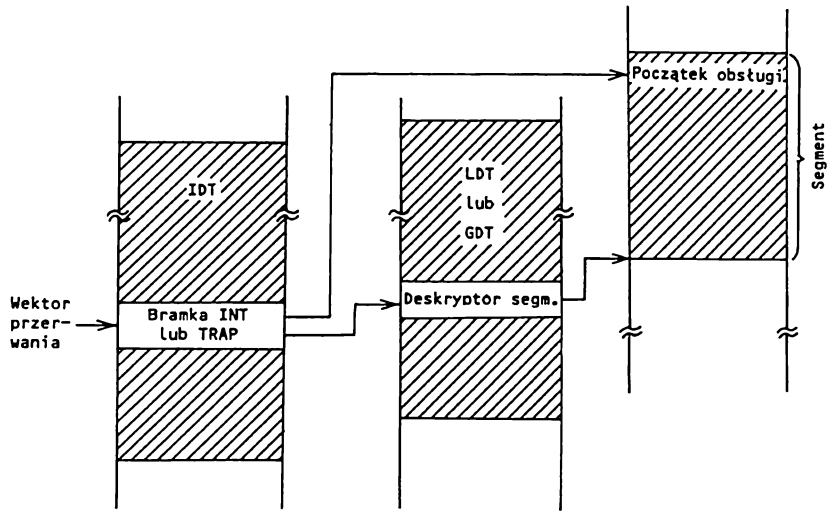


Rys. 4.6. Tablica LDT oraz segmenty zadań dla dwóch zadań w systemie



TGD – ang. *task gate descriptor* – deskryptor bramki zadania
Back Link – selektor poprzedniego zadania

Rys. 4.7. Wymiana zadań na skutek przerwania



Rys. 4.8. Sposób realizacji przerwania bez wymiany zadań

DPL (ang. *descriptor privilege level*) – poziom wynikający z zapisu bajtu dostępu w deskryptorze;

RPL (ang. *requested privilege level*) – poziom "zażądany", wynikający z wartości, którą powinien być załadowany rejestr segmentowy.

Wymiana zadań powoduje wykonanie następujących kroków:

1. Sprawdzenie, czy bieżące zadanie jest uprawnione do przełączenia planowanego zadania. Poziom uprzywilejowania dostępu do danych odgrywa rolę w przypadku rozkazów JMP lub CALL. Poziom uprzywilejowania DPL deskryptora TSS lub bramki zadania musi być niższy (lub równy – tzn. liczbowo większy lub równy) od maksymalnej wartości CPL oraz od poziomu RPL dla selektora bramki. Jeżeli wymianę zadań spowodowało przerwanie zewnętrzne oraz rozkaz IRET, to przełączenie zadań następuje bez uwzględniania poziomu DPL bramki dla docelowego zadania bądź deskryptora TSS;
2. Sprawdzenie, czy deskryptor TSS nowego zadania jest oznaczony jako "obecny" (tzn. rezyduje w pamięci operacyjnej, a nie jedynie na dysku) i ma właściwy rozmiar (limit). Wszystkie błędy zdarzające się do tego momentu dotyczą zadania przerywanego (lub po prostu kończącego pracę). Błędy mogą być obsłużone przez system operacyjny w sposób niewidoczny dla programu użytkownika;
3. Zachowanie stanu bieżącego zadania. W procesorze odnajduje się adres bazowy bieżącego segmentu TSS, przechowywany w rejestrze zadania TR.

Zawartość rejestrów procesora kopiuje się do bieżącego segmentu TSS w sposób pokazany na rys. 4.6. Pole rejestru IP w segmencie TSS wskazuje rozkaz umieszczony w pamięci, po rozkazie wykonanym jako ostatni przed zmianą zadania;

4. Załadowanie rejestru zadania selektorem deskryptora segmentu TSS dla nowego zadania. Deskryptor oznacza się jako aktywny (ang. *busy*), a bit TS w słowie stanu procesora MSW ustawia się w stan 1. Selektor może być albo argumentem rozkazu transmisji danych, albo pobrany z bramki zadania;
5. Załadowanie stanu nowego zadania z jego segmentu TSS i rozpoczęcie lub podjęcie na nowo jego wykonania. Ładuje się przy tym następujące rejestry: LDT, rejestr znaczników, IP, AX, CX, DX, BX, SP, BP, SI, DI, ES, CS, DS. Każdy błąd, który wystąpi na tym etapie przełączania zadań, dotyczy nowego zadania.

Każda *wymiana zadań* powoduje ustawienie znacznika TS w słowie stanu MSW. Znacznik ten wykorzystuje się w oprogramowaniu systemowym, w pracy z koprocesorem (np. arytmetycznym). Znacznik TS sygnalizuje, że stan koprocesora może nie korespondować z bieżącym zadaniem, realizowanym w procesorze 80286. Poziom uprzywilejowania, na którym rozpoczyna się wykonywanie nowego zadania nie jest ani ograniczony, ani narzucony przez poziom zadania realizowanego poprzednio. Ponieważ zadania są izolowane poprzez rozdzielenie przestrzeni adresowej oraz segmenty TSS, a także ponieważ reguły dotyczące przywilejów mogą być zastosowane przy zapobieganiu nieprawidłowemu dostępowi do segmentu TSS, reguł tych nie wykorzystuje się do ograniczania relacji między poziomami CPL poszczególnych zadań. Nowe zadanie rozpoczyna pracę z poziomem uprzywilejowania, określonym przez RPL w selektorze CS, którego wartość ładuje się z segmentu TSS.

Poniżej zestawiono czynności sprawdzające, wykonywane przy przełączaniu zadań (tablica na następnej stronie).

Realizację przełączania zadań w procesorze pracującym w trybie adresowania wirtualnego przedstawimy na przykładzie kilku programów. Załóżmy, że mikrokomputer z procesorem 80286 oraz sterownikiem przerwań 8259 zawiera układ we–wy szeregowych (układ 8251), przyłączony do prostego monitora z klawiaturą oraz układ czasowy (8253), zapewniający realizację przebiegów czasowych w organizacji transmisji we–wy.

W programie *Przykład_1* przedstawimy obsługę klawiatury i ekranu przy założeniu, że procesor 80286 pracuje w trybie adresowania rzeczywistego bez przerwań priorytetowych (obsługa monitora w trybie *polling*). Program pracuje w pętli:
– czyta znak z klawiatury;
– wyświetla odczytany znak na ekranie;
– wraca do początku, czyli czytania znaku z klawiatury.

Po analizie tego prostego przykładu łatwiejsze będzie zrozumienie następnych programów, w których realizuje się to samo zadanie, ale w procesorze 80286 pracującym w trybie adresowania wirtualnego. Jako system uruchomieniowy wykorzystano mikrokomputer klasy IBM PC/AT z dołączonym emulatorem I2ICE.

Test	Opis testu	Błąd	Dotyczy
1	Czy „jest obecny" deskryptor nowego zadania	NP	nowego TSS
2	Czy deskryptor nowego zadania jest oznaczony jako nieaktywny, (ang. *not busy*)	GP	nowego TSS
3	Czy wielkość nowego segmentu TSS jest mniejsza od 43	TS	nowego TSS
	ładowanie wszystkich rejestrów i selektorów		
4	Czy tablica LDT nowego zadania jest prawidłowa (ważna)	TS	nowego TSS
5	Czy tablica LDT nowego zadania „jest obecna" w pamięci operacyjnej	TS	nowego TSS
6	Czy selektor CS jest prawidłowy (ważny)	TS	segmentu programu
7	Czy segment programu „jest obecny"	NP	segmentu programu
8	Czy poziom DPL segmentu programu odpowiada poziomowi RPL dla CS	TS	segmentu programu
9	Czy segment stosu jest prawidłowy (ważny)	GP	segmentu stosu
10	Czy segment stosu „jest obecny"	SF	segment stosu
11	Czy dla segmentu stosu DPL = CPL	SF	segment stosu
12	Czy poziom RPL selektora stosu jest równy poziomowi CPL (RPL = CPL)	GP	segment stosu
13	Czy rejestry DS i ES są prawidłowe	GP	segment
14	Czy segmenty określone przez DS i ES dopuszczają odczyt danych	GP	segment
15	Czy segmenty DS i ES „są obecne"	NP	segment
16	Czy poziom uprzywilejowania DPL segmentów DS, ES jest większy lub równy poziomowi CPL	GP	segment

```
page 60,130
name Przyklad_1
comment ; ***************************************************************
          *     Przykładowy program obsługi szeregowego we-wy klawiatury    *
          *     oraz ekranu monitora (bez systemu MS-DOS).                   *
          *     W programie czyta się znaki z klawiatury i wyświetla je na ekranie.*
          *     Procesor 80286, praca w trybie adresowania rzeczywistego.    *
          ***************************************************************;
dane      segment
          znak      db (?)          ; zmienna wykorzystywana przy przeka-
                                    ; zywaniu znaku wczytanego z klawiatury
dane      ends                      ; i przeznaczonego do wyświetlenia
stos      segment   stack 'stack'
                    dw 20 dup (?)
  szczyt_stosu      label word
stos      ends
; ..........................................................................
; program główny, pracujący w pętli
; ..........................................................................
program   segment
          assume ss:stos, cs:program, ds:program
```

```
start:      mov  ax,stos                ; inicjowanie rejestrów segmentów
            mov  ss,ax                  ; danych i stosu oraz
            mov  ax,program             ; rejestru SP (program przewidziany
            mov  ds,ax                  ; dla systemu bez MS—DOS)
            mov  ax,offset szczyt_stosu
            mov  sp,ax
            call far ptr inicj          ; inicjowanie układów 8251 i 8253
pętla_1:    call far ptr input          ; pobranie znaku z klawiatury
            call far ptr output         ; wyświetlenie znaku na ekranie monitora
            jmp  short petla_1

program     ends
;  ..........................................................................
;  podprogramy obsługi szeregowego we–wy
;  ..........................................................................
sio3da      equ  064h                   ; adres danych w układzie 8251
sio3ct      equ  065h                   ; adres słowa kontrolnego w układzie 8251
tim3        equ  06ah                   ; adres danych w układzie czasowym 8253
timco       equ  06bh                   ; adres słowa kontrolnego w układzie 8253

podprogr    segment
            assume    cs:podprogr, ds:dane, ss:stos
;  ..........................................................................
;  podprogram inicj inicjujący układy: 8251 i 8253
;  ..........................................................................
inicj       proc far
            mov  al,10101010b           ; przygotowanie zerowania
            out  sio3ct,al              ; układu 8253 (SIO–3)
            mov  al,01000000b           ; zerowanie układu 8253 (SIO–3)
            out  sio3ct,al
            mov  al,01111010b           ; wybór trybu pracy układu 8253 (SIO–3)
            out  sio3ct,al
            mov  al,00110111b           ; słowo kontrolne 8253 (SIO–3)
            out  sio3ct,al
            mov  al,10010110b           ; słowo kontrolne 8253 (SIO–3)
            out  timco,al
            mov  al,20h                 ; wartość dzielnika częstotliwości
            out  tim3,al                ; układu 8253 (SIO–3)
            ret
inicj       endp
;  ..........................................................................
;  odczytanie jednego znaku z klawiatury
;  ..........................................................................
input       proc far
pętla_2:    in   al,sio3ct              ; odczytanie słowa stanu
```

```
            and   al,2                  ; RxRDY ?
            jz    short petla_2         ; nie ma znaku
            in    al,sio3da             ; odczytanie znaku
            mov   znak,al               ; zapamiętanie odczytanego znaku
            ret                         ; w zmiennej znak
input       endp
;  ..........................................................................
;  wysłanie jednego znaku na ekran monitora
;  ..........................................................................
output      proc  far
pętla_3:    in    al,sio3ct             ; odczytanie słowa stanu
            and   al,1                  ; TxRDY ?
            jz    short petla_3         ; monitor nie gotowy
            mov   al,znak               ; zmienna znak – na ekran monitora
            out   sio3da,al
            ret
output      endp
podprogr    ends
end         start
```

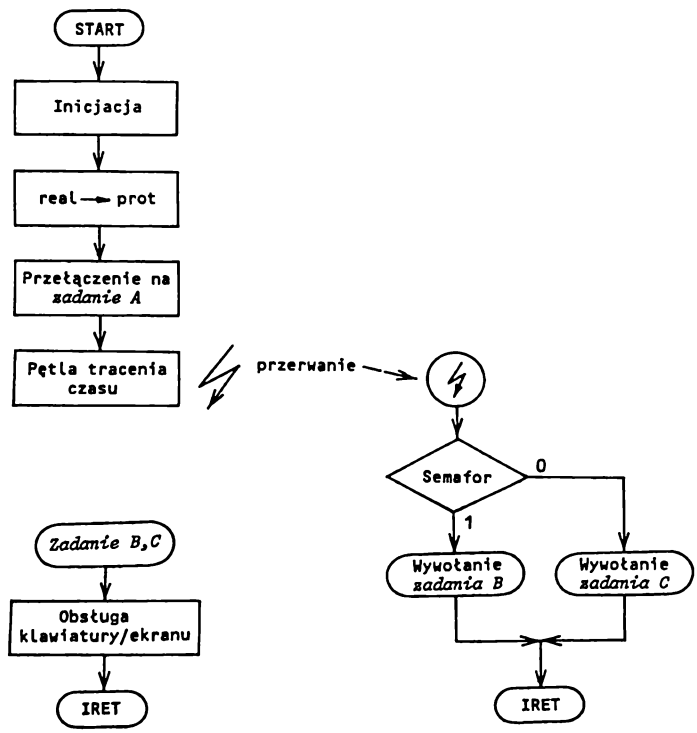
W kolejnym programie realizuje się to samo (jedno) zadanie, a procesor wprowadza się w tryb adresowania wirtualnego. Pobieranie znaków z klawiatury oraz wyświetlanie ich na ekranie odbywa się podobnie, jak w poprzednim przykładzie bez pomocy przerwań.

```
page 60,130
name Przyklad_2
.286p
comment ;  ************************************************************
           *   Przykładowy program obsługi szeregowego we–wy          *
           *   klawiatury i ekranu monitora.                          *
           *   Procesor 80286, praca w trybie adresowania wirtualnego,*
           *   bez przerwań (polling).                                *
           ************************************************************;
stos       segment  stack 'stack'
                    dw 256 dup (?)
    szczyt_stosu    label word
stos       ends
;  ..........................................................................
;  stałe
;  ..........................................................................
cr         equ  0dh
lf         equ  0ah
eos        equ  0ffh                ; znacznik końca łańcucha

sp_rzecz   equ  100h                ; wskaźnik stosu dla trybu adresowania
```

```
                                        ; rzeczywistego
sp_wirt     equ  0h                     ; wskaźnik stosu dla trybu adresowania
                                        ; wirtualnego
; ..........
; inicjacja deskryptorów oraz trybu pracy
; ..........
inicjacja       segment er
                assume ss:stos, ds:program, cs:inicjacja
  adres_GDT     dw   40h                ; wielkość tablicy GDT (limit)
                dw   2800h              ; adres bazowy GDT
                dw   0h
  start:        cli
                mov  ax,stos            ; inicjacja stosu w trybie adresowania
                mov  ss,ax              ; rzeczywistego
                mov  sp,sp_rzecz
                call start1

                ; Część inicjująca programu jest oddalona od adresu, który ustawia
                ; się po zerowaniu (Reset), nie więcej niż o 64 KB. W celu
                ; skorygowania przyjętego przez asembler offsetu
                ; segmentu programu, wykonano rozkaz CALL (krótki),
                ; co spowodowało zapisanie do stosu zawartości rejestru IP.
                ; Wartość tę wykorzystano następnie do obliczenia offsetu
                ; początku segmentu tak, aby właściwie załadować
                ; stałe znajdujące się na początku segmentu.

   start1:  pop  bp                     ; obliczenie offsetu  początku segmentu
            sub  bp,offset start1
            lgdt adres_GDT[bp]
            smsw ax                     ; pobranie słowa stanu procesora
            or   ax,1                   ; ustawianie bitu trybu adresowania wirtualnego
            lmsw ax                     ; przełączenie trybu
            jmp  next                   ; skok w celu opróżnienia kolejki rozkazów

   next:    mov  ax,28h                 ; deskryptor stosu w tablicy GDT (5–ty zapis)
            mov  ss,ax
            mov  sp,sp_wirt
            jmp  far ptr start2

inicjacja   ends
; ..........
; główna część programu
; ..........
program     segment er
            assume ss:stos, ds:program, cs:program
```

```
 komunikat db  cr,lf,'Tryb adresacji wirtualnej',cr,lf,eos
 start2:
; ..............................................................................
; inicjacja układów VLSI
; ..............................................................................
            ;
            ; Inicjowanie układów VLSI przeprowadzono, tak jak w poprzednim
            ; przykładzie
            ; (porównaj podprogram inicj w programie Przyklad_1
            ;
; ..............................................................................
; wyświetlenie komunikatu na ekranie
; ..............................................................................
            lea   ax,komunikat          ; adres tekstu do wyświetlenia
            push  ax                    ; przekazany przez stos do
            call  far ptr wyswietl      ; podprogramu wyświetlenia komunikatu
program     ends
podprogr    segment eo
            assume    ss:stos, ds:podprogr, cs:podprogr
; ..............................................................................
; główna pętla programu, w której realizuje się czytanie z klawiatury i pisanie
; odczytanych znaków na ekranie monitora
; ..............................................................................
  input:    in    al,sio3ct             ; odczytanie słowa stanu
            and   al,2                  ; RxRDY ?
            jz    short input           ; nie ma znaku
            in    al,sio3da             ; odczytanie znaku
            mov   bl,al
   output:  in    al,sio3ct             ; odczytanie słowa stanu
            and   al,1                  ; TxRDY ?
            jz    short output          ; monitor nie gotowy
            mov   al,bl                 ; zmienna znak – na ekran monitora
            out   sio3da,al
            jmp   short input           ; kolejny znak
; ..............................................................................
; podprogram wyswietl – wyprowadzenie komunikatu na ekran
; ..............................................................................
 wyświetl   proc
            push  ax
            push  bx
            push  bp
            mov   bp,sp
            mov   bx,[bp+2*5]           ; parametr podprogramu pobrany ze stosu
            mov   ax,program            ; uaktualnienie rejestru segmentu danych
            mov   ds,ax
```

```
            xor   si,si                    ; zerowanie wskaźnika bufora
pętla:      in    al,sio3ct                ; odczytanie słowa stanu
            and   al,1                     ; TxRDY ?
            jz    petla
            mov   al,komunikat[si]         ; litera do wyświetlenia
            cmp   al,eos                   ; czy koniec tekstu ?
            je    koniec
            out   sio3da,al
            inc   si                       ; następna litera
            jmp   short petla
koniec:     pop   bp
            pop   bx
            pop   ax
            ret   2
wyświetl    endp
podprogr    ends
            end
```



Rys. 4.9. Schemat blokowy programu *Przykład 3*

W ostatnim przykładzie przedstawimy wielozadaniową pracę procesora 80286; rozwinięcie przykładu poprzedniego. W identyczny sposób są tu obsługiwane dwa monitory przyłączone do dwóch szeregowych układów we–wy. Obsługę tę realizuje się w zadaniu *B* oraz *C*. Pracę systemu inicjuje się w zadaniu głównym *A* (w przykładzie jest to zamknięta pętla), przerywanym po przyjęciu przerwania INT 10H.

Na schemacie blokowym z rys. 4.9 przedstawiono główne funkcje tego programu.

```
page 60,130
name Przyklad_3
.286p
comment ; ************************************************************
          *   Przykładowy program obsługi szeregowego we–wy          *
          *   klawiatur i monitorów, praca z przerwaniami.           *
          *   Program obejmuje trzy niezależnie realizowane zadania: *
          *   zadanie A – program główny – inicjowanie pracy systemu,*
          *               a następnie realizowanie "pętli", przerywanej*
          *               zadaniami B i C,                           *
          *   zadanie B – obsługujące monitor 1,                     *
          *   zadanie C – obsługujące monitor 2, identycznie jak zad. B *
          *   Procesor 80286, praca w trybie adresowania wirtualnego *
          ************************************************************
; ..............................................................................
; segment stosu dla trybu adresowania rzeczywistego
; ..............................................................................
stos        segment  stack 'stack'
                     dw 256 dup (?)
   szczyt_stosu      label word
stos        ends
; ..............................................................................
; stałe
; ..............................................................................

cr          equ  0dh
lf          equ  0ah
eos         equ  0ffh              ; znacznik końca łańcucha
sp_rzecz    equ  100h              ; wskaźnik stosu dla trybu adresowania
                                   ; rzeczywistego
sp_wirt     equ  0h                ; wskaźnik stosu dla trybu adresowania
                                   ; wirtualnego
; ..............................................................................
; symbole globalne
; ..............................................................................
public      start, przerwanie                    ; symbole programu Builder
extrn       task_A:far, task_B:far, task_C:far    ; symbole bramek zadań
```

```
;  .....
;  dane globalne
;  .....
dane_gl     segment  rw
            semafor  db 0h
dane_gl     ends
;  .....
;  inicjacja programu oraz trybu pracy
;  .....
program_inic    segment er
                assume  ss:stos, ds:program_inic, cs:program_inic
  adres_GDT     dw   70h            ; wielkość tablicy GDT (limit)
                dw   2800h          ; adres bazowy GDT
                dw   0h
  adres_IDT     dw   0f7h           ; wielkość tablicy IDT
                dw   50h            ; adres bazowy IDT
                dw   0h
  start:        cli
                mov  ax,stos        ; inicjacja stosu w trybie adresowania
                mov  ss,ax          ; rzeczywistego
                mov  sp,sp_rzecz
                call start1
  start1:       pop  bp             ; obliczenie off setu początku segmentu
                sub  bp,offset start1 ; (zobacz komentarz do Przyklad_2)
                lgdt adres_GDT[bp]
                lidt adres_IDT[bp]
                smsw ax             ; pobranie słowa stanu procesora
                or   ax,1           ; ustawianie bitu trybu adresowania wirt.
                lmsw ax             ; przełączenie trybu
                jmp  next           ; skok w celu opróżnienia kolejki rozkazów
  next:         mov  ax,50h         ; deskryptor stosu w tablicy GDT (10–ty zapis)
                mov  ss,ax
                mov  ax,18h         ; deskryptor LDT w tablicy GDT (3–ci zapis)
                lldt ax             ; ładowanie rejestru LDTR
                mov  sp,sp_wirt     ; inicjacja wskaźnika stosu
                mov  ax,20h         ; 4–ty zapis w tablicy GDT jest
                                    ; tzw. "dummy–task–state–segment"
                ltr  ax             ; Jest on potrzebny
                                    ; przy pierwszej
                                    ; wymianie zadań do składowania aktual-
                                    ; nej zawartości rejestrów
                jmp  far ptr task_a ; skok do bramki zadania A (wymiana
                                    ; zadań)

program_inic    ends
```

```
; .....
; zadanie A – program główny
; .....
task_A_code     segment er
                assume ss:stos, ds:task_A_code, cs:task_A_code
                public start2              ; symbole dla programu Builder
  komunikat     db   cr,lf,'Tryb adresowania wirtualnego',cr,lf,eos
  start2:
; .....
; inicjacja układów VLSI
; .....
                ;
                ; Inicjowanie układów VLSI przeprowadzono analogicznie, jak
                ; w poprzednich przykładach, dla dwóch szeregowych
                ; we–wy (porównaj podprogram inicj w programie Przyklad_1)
                ;
                  mov  al,0feh             ; ustawienie maski przerwań, w celu
                  out  dx,al               ; odblokowania tylko INT 10H
                  lea  ax,komunikat        ; adres efektywny komunikatu do AX
                  push ax                  ; przekazanie parametru przez stos
                  call wyswietl
                  sti
; .....
; pętla zamknięta programu głównego
; .....
  pętla1:       nop
                sti
                int  10h
                jmp  short petla1
; .....
; podprogram – wyświetlenie komunikatu na ekranie
; .....
  wyświetl      proc near
                push ax
                push bx
                push bp
                mov  bp,sp
                mov  bx,[bp+2*5]
                mov  ax,task_A_code
                mov  ds,ax
  pętla2:       in   al,sio3ct
                and  al,1
                jz   petla2
                mov  al,[bx]
                cmp  al,eos
```

```
                je    koniec_wysw
                out   sio3da,al
                inc   bx
                jmp   short petla2
koniec_wysw:    pop   bp
                pop   bx
                pop   ax
                ret   2
  wyświetl      endp
task_A_code     ends
;  ..........
;  obsługa przerwania int 10h
;  ..........
przerw_seg      segment er
                assume ss:stos, ds:dane_gl, cs:przerw_seg
  przerwanie    proc far
                cli
                push ax
                mov  ax,dane_gl
                mov  ds,ax
                mov  al,semafor
                or   al,al              ; ustawienie znacznika ZF
                jz   prz1               ; jeżeli semafor = 1, to zadanie B
                                        ; w przeciwnym wypadku zadanie C
                call far ptr task_B     ; skok do bramki zadania utworzonego
                                        ; przez program Builder (wymiana zadań)
                ; Uwaga ! rozkaz CALL echo_1 nie spowoduje wymiany zadań,
                ;          a jedynie skok międzysegmentowy
                jmp  prz2               ; koniec, nic więcej do roboty !
  prz1:         call far ptr task_C     ; skok do bramki zadania
  prz2:         pop  ax
                iret                    ; powrót do "zamkniętej pętli"
 przerwanie     endp
przerw_seg      ends
;  ..........
;  zadanie B – obsługuje SIO2
;  ..........
task_B_code     segment er
                assume cs:task_B_code, ss:stos, ds:dane_gl
                public echo_1           ; symbol dla programu Builder (adres
                                        ; startowy) ładowanie rejestrów
                                        ; segmentowych w programie nie jest
                                        ; potrzebne, będą one automatycznie
                                        ; załadowane przy wymianie zadań
                                        ; z segmentu TSS
```

```
  echo_1        proc far
  input_1:      in    al,sio2ct          ; odczytanie słowa stanu
                and   al,2               ; RxRDY ?
                jz    short koniec_B     ; nie ma znaku
                in    al,sio2da          ; odczytanie znaku
                mov   bl,al
  output_1:     in    al,sio2ct          ; odczytanie słowa stanu
                and   al,1               ; TxRDY ?
                jz    short output_1     ; monitor nie gotowy
                mov   al,bl              ; zmienna znak – na ekran monitora
                out   sio2da,al
  koniec_B:     mov   al,semafor         ; zmiana stanu semafora taka, aby przy
                                         ; następnym przerwaniu było uruchomione
                not   al                 ; inne zadanie
                mov   semafor,al
                iret                     ; przy wywołaniu zadania rozkazem
                                         ; FAR–CALL, powrót do zadania przerwa-
                                         ; nego musi odbywać się za pośrednictwem
                                         ; rozkazu IRET. Wykonanie rozkazu IRET
                                         ; spowoduje testowanie znacznika NT
                                         ; w słowie MSW i przejście do obsługiwania
                                         ; przełączania zadań.
                jmp   input_1            ; przy ponownym uruchomieniu zadania
                                         ; będzie ono kontynuowane od tego miejsca
                                         ; (wynika to z zawartości rejestrów
                                         ; zachowanych w segmencie TSS) z tego
                                         ; względu skok do początku obsługiwania
                                         ; zadania
  echo_1        endp
task_B_codeends
;  .....
;  zadanie C     – obsługuje SIO3
;                ! zadanie C jest zbudowane identycznie jak zadanie B!
;                ! z tego powodu komentarz jest uproszczony !
;  .....
task_C_code     segment er
                assume cs:task_C_code, ss:stos, ds:dane_gl
                public echo_2            ; symbol dla programu Builder
                                         ; (adres startowy)
  echo_2        proc far
  input_2:      in    al,sio3ct          ; odczytanie słowa stanu
                and   al,2               ; RxRDY ?
                jz    short koniec_C     ; nie ma znaku
                in    al,sio3da          ; odczytanie znaku
                mov   bl,al
```

```
output_2:   in    al,sio3ct          ; odczytanie słowa stanu
            and   al,1               ; TxRDY ?
            jz    short output_2     ; monitor nie gotowy
            mov   al,bl              ; zmienna znak – na ekran monitora
            out   sio3da,al
koniec_C:   mov   al,semafor         ; zmiana stanu semafora
            not   al
            mov   semafor,al
            iret
            jmp   input_2
echo_2      endp
task_C_code ends
            end
```

4.3. Wielozadaniowy system operacyjny OS/2

W mikrokomputerach klasy IBM PC/AT, pracujących pod kontrolą systemu operacyjnego MS–DOS, najistotniejsze oraz najciekawsze cechy mikroprocesora 80286 są niewykorzystywane – procesor pracuje w trybie adresowania rzeczywistego i z punktu widzenia użytkownika różni się od mikroprocesorów 8086/8088 jedynie szybkością i nieco bogatszą listą rozkazów. Gwałtowny rozwój technologii wytwarzania układów półprzewodnikowych przyczynił się do szybkiego wprowadzania na rynek kolejnych mikroprocesorów firmy Intel, o coraz większych możliwościach. Mikroprocesor 8088, wyprodukowany w 1981r. i zastosowany jako jednostka centralna w pierwszym mikrokomputerze IBM PC, pracował przy częstotliwości zegara 4.77 MHz i zapewniał bezpośrednie adresowanie 1 MB pamięci. W mikroprocesorze 80286 (1984r.) obszar pamięci operacyjnej wynosi już 16 MB, a w mikroprocesorze 80386 (1986r.) – 4 GB pamięci i częstotliwość zegara – 20 MHz[1]. Niestety, twórcy oprogramowania systemowego mikrokomputerów, opartych na tych procesorach, nie nadążają za tak szybkim rozwojem.

Użytkownicy "super AT" z bardzo szybkimi procesorami 80386, pracując z systemem operacyjnym MS–DOS, nadal mają do dyspozycji tylko 640 KB pamięci RAM. Ewentualne rozszerzenie pamięci uzyskuje się, stosując wirtualny dysk lub pamięć przyspieszającą niektóre operacje dyskowe. W firmie Microsoft, (dostawca systemu MS–DOS), zareagowano na potrzeby użytkowników oraz zwiększone możliwości sprzętu po raz pierwszy w 1985 roku, wypuszczając na rynek program o nazwie MS–Windows, którego wersja 2.0 jest graficzną nakładką na system MS–DOS (technika okienek, ikony) i umożliwia symulowanie wielozadaniowości.

Program MS–Windows to bezpośrednia zapowiedź nowego systemu operacyjnego OS/2 w wersji 1.1 (*Presentation Manager*) [16]. Następny, pośredni krok to

---

[1] Dane te szybko się dezaktualizują. Obecnie są dostępne procesory pracujące przy większych częstotliwościach, wyprodukowano już procesor 80486.

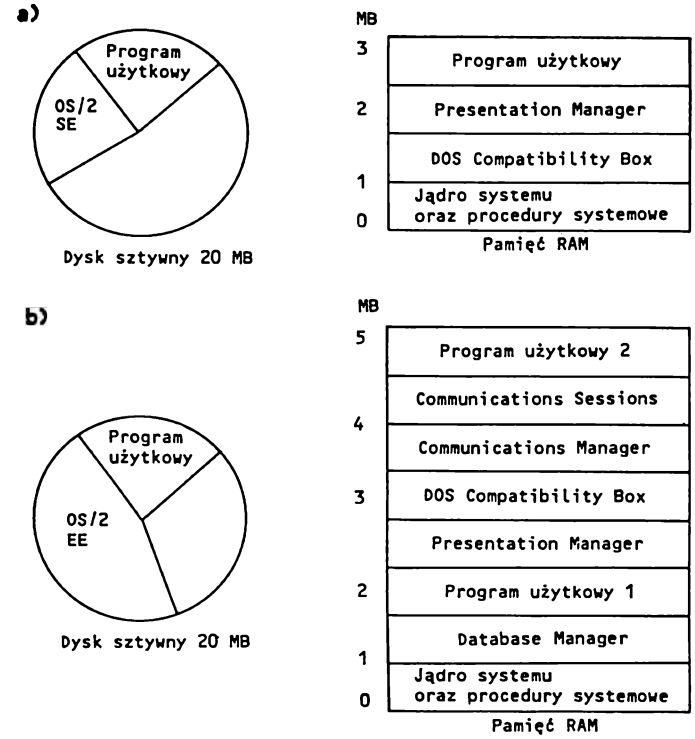
Windows–386 dla komputerów z procesorem 80386. Program Windows–386 umożliwia realizację wielozadaniowości przez utworzenie do ośmiu wirtualnych komputerów, pracujących pod systemem MS–DOS. Wykorzystuje się tutaj pracę mikroprocesora 80386 w trybie wirtualnym–86 (*virtual 8086 mode*), z równoległą realizacją konwencjonalnych programów, przygotowanych dla systemu MS–DOS. Możliwości takich nie ma w systemie operacyjnym OS/2, który wykorzystuje jedynie tryby adresowania rzeczywistego i wirtualnego procesora 80286. Pracując z programem Windows–386, w dalszym ciągu mamy do dyspozycji tylko 640 KB pamięci RAM dla pojedynczego zadania, jest jednak możliwa automatyczna wymiana danych między poszczególnymi zadaniami.

Wprowadzenie systemu operacyjnego OS/2, którego główną cechą jest wielozadaniowość (w tym wielozadaniowość wewnątrz jednego zadania), nie jest oczywiście jednoznaczne z natychmiastową możliwością przeniesienia dotychczasowego, bardzo bogatego oprogramowania użytkowego pracującego pod systemem MS–DOS. Proces ten jest złożony. Aby zapewnić zgodność systemu MS–DOS, z dotychczasowym środowiskiem, system OS/2 umożliwia wykorzystanie tzw. *compatibility–box*, który umożliwia z kolei pracę procesora 80286 w trybie adresowania rzeczywistego. System DOS–box, zwany też *family–box*, symuluje środowisko systemu MS–DOS, zapewniając jednocześnie realizację większości dotychczasowych programów. Oczywiście nie ma wówczas możliwości realizowania wielozadaniowości, a przejście do trybu adresowania wirtualnego powoduje przerwanie realizowanego zadania. Ponowne przełączenie wymusza powrót do przerwanego zadania. W tym czasie jednak mogą być wykonywane "w tle" operacje uaktywnione w trybie adresowania wirtualnego. Zastosowanie systemu DOS–box nie wpływa na przerwanie wielozadaniowości. Trzeba tutaj dodatkowo podkreślić, że niektóre programy są wykonywane szybciej pod systemem MS–DOS 3.3 aniżeli za pośrednictwem DOS–box systemu OS/2. Przyczyna tkwi przede wszystkim w architekturze samego procesora 80286. Przełączanie z trybu adresowania wirtualnego do rzeczywistego wiąże się z wykonaniem (każdorazowo) tzw. *"ciepłego startu"*, czyli wyzerowaniem procesora. Podczas "równoczesnej" realizacji zadań w trybie adresowania wirtualnego oraz za pośrednictwem DOS–box, procesor musi być ciągle przełączany, a tym samym i zerowany.

System operacyjny OS/2 jest całkowicie nowym systemem, umożliwiającym zdefiniowanie (przez jednego użytkownika) i równoczesne wykonywanie do 12 zadań. Po raz pierwszy wykorzystuje on tryb adresowania wirtualnego[1] procesorów 80286 i 80386. Dzięki systemowi OS/2 przełamano w końcu barierę 640 KB pamięci RAM dla użytkownika. Nowy system umożliwia bezpośrednie adresowanie do 16 MB pamięci, a wirtualne (wykorzystując dysk) do 1GB (1000 MB). W koncepcji pamięci wirtualnej zakłada się składowanie aktualnie niepotrzebnych części programu na dysku stałym. Zadaniem systemu operacyjnego jest takie gospodarowanie pamięcią, aby niezbędne części programu były wprowadzane do pamięci operacyjnej we właściwym momencie. Pamięć wirtualna, w porównaniu z pamięcią operacyjną, umożliwia realizację bardziej rozbudowanych programów.

---

1 Mówimy o systemach firm IBM i Microsoft. W trybie adresowania wirtualnego procesory 80286 i 80386 pracują także pod systemem operacyjnym UNIX.



Rys. 4.10. Zapotrzebowanie na pamięć systemu *OS/2*: a) *Standard Edition* – przy pamięci operacyjnej rzędu 3 MB niewiele miejsca pozostaje dla programów użytkowych, co doprowadzi do częstych komunikacji z dyskiem; b) *Extended Edition* – praca z kilkoma programami użytkowymi wymaga zastosowania dysku o pojemności większej niż 20 MB

System operacyjny OS/2 jest wspólnym produktem firm IBM i Microsoft. Pierwszą dostępną na rynku standardową wersję systemu (*Standard Edition 1.0* – rys. 4.10) charakteryzuje

– zarządzanie pamięcią operacyjną do 16 MB;
– równoczesne realizowanie wielu zadań (do 12 sesji);
– komunikowanie się między procesami – wielozadaniowość w obrębie jednego zadania;
– obecność selektora programów;
– dostępność podpowiedzi (*on–line*) o funkcjach systemu (tzw. *help*).

Kolejną wersję *1.1* uzupełniono

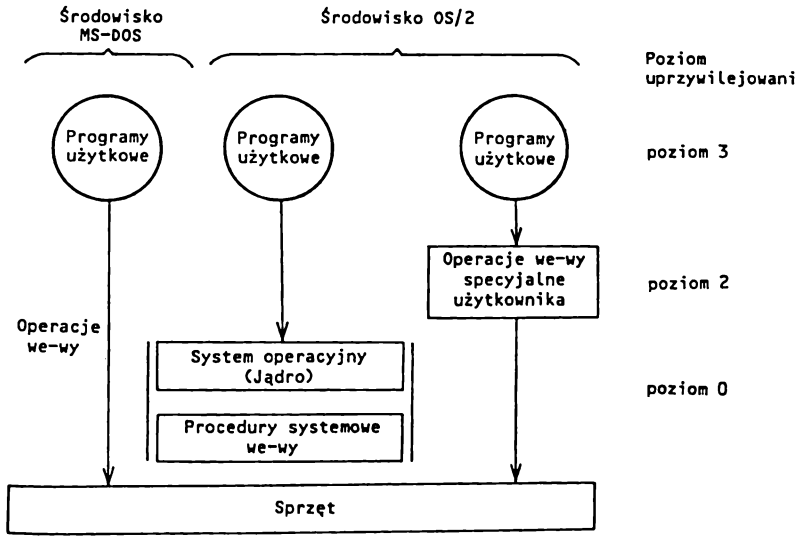
– graficznym sposobem komunikowania się z użytkownikiem na wzór programu Windows 2.0 (*Presentation Manager*);
– edytorem systemowym (następca programu EDLIN);
– podziałem stałego dysku większego niż 32 MB.

Rozszerzoną wersję systemu OS/2 (*Extended Edition 1.0* – rys. 4.10b oraz wersję *1.1*) rozbudowano o moduł

– zarządzający komunikacją z komputerami podrzędnymi (*Communications Manager*);
– zarządzający zbiorami danych (*Database Manager*);
– sterujący siecią lokalną (*LAN Server*).

Przy pracy wielozadaniowej konieczna jest właściwa gospodarka pamięcią operacyjną tzn. taka, aby nie spowodować kolizji aktualnie wykonywanych programów, które w całości lub w części wczytuje się z pamięci dyskowych do pamięci operacyjnej, optymalnie wykorzystując dostępne zasoby pamięci. Jak wiadomo z poprzednich rozdziałów, mikroprocesor 80286 pracujący w trybie adresowania wirtualnego wyposażono w mechanizmy sprzętowe, za pomocą których można gospodarować segmentami pamięci w sposób zabezpieczający programy i dane między sobą, a także można uniemożliwić niszczącą ingerencję w system operacyjny.

Jest rzeczą zrozumiałą, że sposób zarządzania pamięcią operacyjną wyklucza możliwość pisania programów wprost do pamięci operacyjnej. Każdy program będzie przypisany do właściwego segmentu pamięci zgodnie z jego potrzebami oraz w sposób, wynikający z aktualnego stanu systemu. Fizyczny adres tego segmentu nie jest programowi "znany". Można powiedzieć, że w ogóle dostęp programu użytkowego bezpośrednio do sprzętu jest znacznie ograniczony (rys. 4.11). Dotyczy to nie tylko pamięci operacyjnej oraz stałego dysku, mającego decydujące znaczenie dla realizacji koncepcji pamięci wirtualnej, ale także pozostałych urządzeń peryferyjnych.



Rys. 4.11. Dostęp programu użytkowego do sprzętu w systemach *MS-DOS* oraz *OS/2*

Mechanizmy ochrony, wykorzystywane w systemie OS/2, zapewniają zabezpieczenie przed korzystaniem w programach użytkowych z nieprzydzielonych obszarów pamięci. Nie jest możliwe, tak jak to było w systemie MS–DOS, przeprowadzanie operacji we–wy bezpośrednio na poziomie sprzętu. Mechanizm "przywilejów" systemu OS/2 zapewnia bezpieczeństwo danych poszczególnych procesów i nie

dopuszcza do kolizji przy operacjach we–wy. Nie wszystkie rozkazy procesora 80286 są dopuszczalne na wszystkich poziomach uprzywilejowania. W szczególności, w programie użytkowym, którego poziom uprzywilejowania jest najniższy, nie dopuszcza się wielu rozkazów. Rozkazy, takie jak: HALT, IRET, POPF mogą być wykonywane wyłącznie na najwyższym (zerowym) poziomie uprzywilejowania, zarezerwowanym dla jądra systemu operacyjnego. Inne, jak: IN, INW, OUT, OUTW, STI, CLI oraz LOCK są akceptowane na poziomie uprzywilejowania właściwym dla operacji we–wy. Innymi słowy, przy wykonywaniu operacji we–wy są wymagane odpowiednie procedury systemowe i nie można ich realizować bezpośrednio w programie użytkowym.

Wyróżnimy trzy typy *programów użytkowych*, pracujących pod systemem OS/2

1. Programy przeniesione z systemu MS–DOS, uruchamiane we wspomnianym wyżej *Compatibility–Box*. Trzeba tutaj podkreślić, że nie wszystkie programy można w prosty sposób przenieść do nowego systemu. Dotyczy to przede wszystkim programów uwarunkowanych czasowo, jak np. programy komunikacyjne, niektóre zastosowania sieciowe, a także programy z wykorzystaniem absolutnego zapisu na dysk (np. *Framework*). Te programy należy napisać na nowo.
2. Druga grupa programów to tzw. *Family API–Programs* (API – ang. *aplication program interface*). Programy tej grupy zastosowano jako pierwsze pod systemem OS/2. Nie korzysta się w nich wprawdzie ze wszystkich możliwości jakie oferuje nowy system operacyjny, ale można je wykonywać zarówno pod systemem MS–DOS, jak i OS/2 w trybie adresowania wirtualnego. W programach tych, przy wykorzystaniu podzbioru możliwości systemu OS/2, jest akceptowana wielozadaniowość, lecz nie wewnątrz jednego zadania. Można przekroczyć granicę 640 KB pamięci RAM i używać pełnego zakresu adresowania systemu OS/2 (16 MB).
3. Trzecia grupa programów, niewątpliwie najbardziej interesująca, to programy, w których wykorzystuje się wszystkie możliwości systemu OS/2. W chwili, gdy piszemy te słowa (grudzień 1988) oczekuje się, że programy tej grupy będą wprowadzone na rynek po 1990 r.

Możliwości systemu OS/2, wykorzystywane w pełni w ostatniej wymienionej grupie programów użytkowych, to przede wszystkim wspomniana już wielozadaniowość w obrębie jednego zadania[1] (programu), komunikacja między procesami, czy wreszcie rozbudowane funkcje uzależnień czasowych. Trudno obecnie przewidzieć jak mocno fantazja i wyobraźnia programistów wpłynie na wykorzystanie tych możliwości i w związku z tym, jakie nowe programy pojawią się na rynku.

Przykładem wielozadaniowości w obrębie zadania mogłaby być praca z procesorem tekstu: w trakcie edycji tekstu w programie jest sprawdzana jego poprawność ortograficzna.

W oprogramowywaniu procesora 80286 pracującego w trybie adresowania wirtualnego (na poziomie systemowym) przed programistą postawiono olbrzymie wymagania, znacznie przewyższające te, które znamy z procesorów 8086/8088 oraz 80286 pracującego w trybie adresowania rzeczywistego. Tłumaczy to tak znaczne

[1] Możliwa jest drzewiasta struktura zadań.

opóźnienie w pojawieniu się ogólnie dostępnego oprogramowania systemowego, wykorzystującego wszystkie możliwości tego 16–bitowego mikroprocesora.

Programista, który tworzy oprogramowanie użytkowe pod systemem operacyjnym OS/2, korzysta z wielu jego nowych możliwości. System ten zapewnia łatwe i sprawne uruchomienie programu.

Należy w tym miejscu raz jeszcze przypomnieć, że programu wynikowego uzyskanego w mikrokomputerze z systemem operacyjnym MS–DOS nie można realizować pod systemem OS/2 (i odwrotnie). Przy próbie uruchomienia takiego programu nastąpi wyświetlenie komunikatu o błędzie

SYS0193: C:pr.exe cannot be run in IBM Operating System/2 mode

Z problemem tym może się spotkać użytkownik, który na dysku stałym ma zainstalowane oba systemy operacyjne. Ponieważ struktura katalogów jest identyczna w obu systemach, a składnia zleceń podobna, mniej uważny użytkownik może wręcz zapomnieć pod jakim systemem aktualnie pracuje. Istotną różnicą na poziomie programów źródłowych, przygotowanych z myślą o pracy pod jednym z tych dwóch systemów, wynikającą ze wspomnianego wcześniej oddalenia użytkownika od poziomu sprzętu w systemie OS/2, jest niedostępność przerwania INT 21H w systemie OS/2, poprzez które są wywoływane procedury systemu MS–DOS. Wywołanie systemowych procedur systemu OS/2 realizuje się rozkazem CALL. Istota systemu OS/2–API tkwi w tym, że procedury systemowe można wywoływać bezpośrednio, umieszczając ich nazwę w programie napisanym w języku wysokiego poziomu. Aby lepiej to przedstawić, trzeba w dalszych rozważaniach zejść na poziom języka asemblera. Na tym poziomie należy wymienić 6 podstawowych reguł obowiązujących programistę

- wszystkie parametry niezbędne w procedurze systemowej należy przekazywać przez stos. Dzięki rozszerzonym opcjom rozkazu PUSH w procesorach 80186 i 80286, można się obejść bez pośrednictwa rejestrów, także w przypadku parametrów będących stałymi;
- po wykonaniu każdej procedury systemowej za pośrednictwem rejestru AX jest przekazywany pewien kod. Przy poprawnym wykonaniu, kod jest równy zero;
- wszystkie procedury wywołuje się rozkazem FAR CALL;
- parametry wywołania umieszczone na stosie są z niego automatycznie usuwane w systemie OS/2;
- parametry wyjściowe procedur są wpisywane do pamięci w obszar zadeklarowany przed wywołaniem procedury. Adres tego obszaru w formacie "selektor:offset" jest przekazywany także za pośrednictwem stosu. Dzięki temu, niezależnie od rodzaju procesora (80286 lub 80386) oraz tzw. *modelu systemu* (ang. *memory model*), określającego wielkość dostępnej pamięci operacyjnej, format danych wejściowych i wyjściowych procedur systemowych może być jednolity;
- wszystkie nazwy, którymi wywołuje się procedury systemowe należy pisać dużymi literami. W niektórych asemblerach bądź kompilatorach następuje automatyczna konwersja małych liter na duże – wtedy oczywiście w programie źródłowym warunek ten nie musi być spełniony.

Ważniejsze procedury systemowe systemu OS/2 wywoływane z programu użytkowego zebrano poniżej.

Funkcja OS/2		Odpowiednik funkcji w MS-DOS	Znaczenie
DosBeep			generowanie sygnału dźwiękowego
DosChDir		INT 21H, 3BH	zmiana aktualnego katalogu
DosChgFilePtr		INT 21H, 42H	zmiana wskaźnika zapisu/odczytu
DosClose		INT 21H, 3EH	zamknięcie zbioru
DosDelete		INT 21H, 41H	usunięcie zbioru
DosDevConfig			stwierdzenie konfiguracji systemu
DosDevIOCtl	*	INT 21H, 44H	sterowanie urządzeń znakowych
DosDupHandle		INT 21H, 45H	utworzenie 2–go kanału logicznego
DosError	*		komunikat o błędzie
DosExit		INT 21H, 4CH	zakończenie programu użytkownika
DosFileLocks	*	INT 21H, 5CH	zajęcie/zwolnienie głowicy zapisującej
DosFindClose	*		zamknięcie handlera katalogu
DosFindFirst	*	INT 21H, 4EH	poszukiwanie 1–go zbioru w katal.
DosFindNext	*	INT 21H, 4FH	poszukiwanie kolejnego zbioru
DosGetDateTime		INT 21H, 2AH,2CH	odczyt. daty i czasu systemowego
DosMkDir		INT 21H, 39H	utworzenie nowego katalogu
DosMove		INT 21H, 56H	zmiana nazwy zbioru
DosNewSize			zmiana wielkości zbioru
DosOpen	*	INT 21H, 3DH	otwarcie istniejącego zbioru
DosQCurDir		INT 21H, 47H	określenie wybranego katalogu
DosQCurDisk		INT 21H, 19H	test, który mechanizm wybrano
DosQFHandState			test stanu kanału dla zbioru
DosQFSInfo		INT 21H, 36H	informacja o wolnych obszarach dysku
DosQFileInfo		INT 21H, 57H	informacja o zbiorze na dysku
DosQFileMode		INT 21H, 43H	pobranie/zmiana atrybutów zbioru
DosQVerify		INT 21H, 54H	pobranie znacznika weryfikacji
DosRead		INT 21H, 3FH	odczyt. danych ze zbioru (urządzenia)
DosRmDir		INT 21H, 3AH	usunięcie pustego katalogu
DosSelectDisk		INT 21H, 0EH	wybór mechanizmu dyskowego
DosSetFileInfo		INT 21H, 57H	data/godzina zapisu do zbioru
DosSetVec		INT 21H, 25H	wpisanie wektora przerwania
DosSetVerify		INT 21H, 2EH	zmiana stanu wskaźnika weryfikacji
DosWrite		INT 21H, 40H	zapis. danych do zbioru (urządzenia)
KbdCharIn		INT 21H, 07H	odczyt. znaku z klawiatury bez echa
KbdFlushBuffer			zerowanie bufora klawiatury
KbdGetStatus		INT 16H, 00H	stwierdzenie stanu klawiatury
KbdPeek	*	INT 16H, 00H	pobranie 'Scan Code' znaku
KbdSetStatus			ustawienie stan klawiatury
KbdStringIn		INT 21H, 0AH	odczyt. z klawiat. łańcucha znaków
VioGetBuf			przydziel. log. bufora dla monitora
VioGerCurPos		INT 10H, 03H	odczytanie pozycji kursora
VioGetCurType			określenie typu kursora
VioGetMode		INT 10H, 0FH	określenie trybu pracy monitora

VioGetPhysBuf		określ.fiz.adr. bufora dla monitora
VioReadCellStr	INT 10H, 08H	odczyt. znaku i jego atrybutu
VioReadCharStr		odczyt. łańcucha znaków
VioScrollDn	INT 10H, 07H	przewijanie ekranu w dół
VioScrollLf		przewijanie ekranu w lewo
VioScrollRt		przewijanie ekranu w prawo
VioScrollUp	INT 10H, 06H	przewijanie ekranu w górę
VioSetCurPos	INT 10H, 02H	ustawienie pozycji kursora
VioSetCurType	INT 10H, 01H	definiowanie rozmiaru kursora
VioSetMode	INT 10H, 00H	wybór trybu pracy monitora
VioShowBuf		wskazanie logicznego bufora dla monitora
VioWrtCellStr	INT 10H, 09H	zapis. znaku i atrybutu
VioWrtCharStr	INT 10H, OAH	zapis. znaku na ekranie bez atrybutu
VioWrtCharStrAtt		zapis. znaku i atrybutu na ekranie
VioWrtNAttr		zapis. atrybutu 'N'
VioWrtCell		zapis. na ekranie 'N' wierszy
VioWrtNChar		zapis. na ekranie 'N' znaków
VioWrtTTY	INT 21H, 02H	dalekopisowy tryb pracy monitora

* – zastosowanie funkcji ograniczone do *"Family Application"*

Stosowanie powyższych funkcji zilustrowano na przykładzie procedury DosRead. W języku C procedurę można wywołać podobnie, jak inne funkcje dostępne w tym języku

DOSREAD (kanał,bufor,wielkość,liczba_bajtów)

gdzie poszczególne parametry oznaczają:

kanał – numer kanału logicznego urządzenia;
bufor – bufor dla odczytanych danych;
wielkość – wielkość bufora, liczba bajtów do odczytania;
liczba_bajtów – ilość rzeczywiście odczytanych bajtów.

Przy programowaniu w asemblerze wywołanie powyższej procedury może mieć postać

```
push kanal          ; 16-bitowy numer kanału logicznego
push bufor_seg      ; 32-bitowy adres bufora dla odczytanych
push bufor_off      ; danych: segment i offeset
push wielkosc       ; 16-bitowa liczba: wielkość bufora danych
push l_b_seg        ; 32-bitowy adres dla zmiennej wyjściowej
push l_b_off        ; informującej o liczbie przeczytanych bajtów
CALL DOSREAD
and ax,ax           ; ustaw znacznik zera ZF, jeśli
                    ; realizacja procedury była bezbłędna
```

Używane w programie nazwy procedur systemowych należy zadeklarować jako zewnętrzne dyrektywą EXTRN. Odpowiednie adresy tych podprogramów można odnaleźć w programie LINK w bibliotece o nazwie DOSCALLS.LIB w trakcie tworzenia zbioru z rozszerzeniem EXE. Warto jeszcze wspomnieć o tym, iż program ładujący systemu OS/2 działa inaczej aniżeli analogiczny program systemu MS-DOS.

Jeżeli w programie segmenty danych będą zgrupowane dyrektywą asemblera GROUP z nazwą DGROUP, to w programie tym nie trzeba już ładować rejestru DS rozkazami przesłania (np. MOV). Automatycznie będą ustawione następujące rejestry:

AX – adresu segmentu zmiennych systemowych (ang. *environment–blocks*);
BX – wskaźnika linii zlecenia w obszarze zmiennych systemowych;
DS – adresu początku segmentu danych;
CS:IP – adresu wejścia do programu;
SS:SP – wskaźnika początku stosu.

5. Przykładowe konfiguracje mikrokomputerów

Mikrokomputery klasy IBM PC/XT/AT, a także system PS/2 to z założenia mikrokomputery uniwersalne. Oznacza to, że przewidywane zastosowanie w stosunkowo małym stopniu wpływa na konfigurację sprzętową oraz, że większość dostępnego na rynkach światowych oprogramowania można wykorzystać w sprzęcie z różnym zestawem podstawowych podzespołów (pamięci dyskowe o różnej wielkości, pakiety graficzne, wielkość pamięci RAM, typ zastosowanej drukarki itp.). Wymaganiem, najczęściej stawianym przez firmy oferujące nowoczesne oprogramowanie, jest minimalna wielkość pamięci RAM (256 KB lub 640 KB) oraz obecność w systemie dysku stałego typu *Winchester*[1]. Osobnym zagadnieniem jest konfigurowanie systemu mikrokomputerowego, opartego na mikrokoputerze klasy IBM PC/XT/AT, przeznaczonego do bardzo specjalistycznych prac, jak sterowanie prostymi obiektami technologicznymi (np. w laboratorium) bądź do współpracy z innymi specjalistycznymi urządzeniami nie komunikującymi się z komputerem przez standardowe układy we–wy; ale i wtedy podstawowe pakiety systemu pozostają uniwersalne, a wymaganą specjalizację osiąga się, stosując pakiety dodatkowe.

Podstawowy zestaw dla konfiguracji uniwersalnej, o znaczeniu już nieco historycznym, jest następujący:[2]

a) dla komputera klasy XT

– procesor 8088, 10/4.77 MHz (tzw. *turbo* – przełączalny);

---

[1] Oprogramowanie, wprowadzane obecnie na rynki komputerowe, jest w większości tworzone w językach wysokiego poziomu, bez przywiązywania zbyt dużej wagi do racjonalnej gospodarki pamięcią komputera.

[2] Zestaw taki jest nadal oferowany przez prawie wszystkie (najczęściej mniej renomowane) firmy, reklamujące funkcjonalnie pełny zestaw za możliwie najniższą cenę. Trzeba także podkreślić, że zestawy oferowane przez różnych dostawców jako zgodne z komputerami IBM PC/XT/AT nie zawsze odpowiadają strukturom oryginałów. Przykładowo, oryginalnego mikrokomputera IBM XT nie sprzedaje się bez sztywnego dysku stałego typu *Winchester*.

– 256 KB pamięci RAM (z podstawkami na płycie głównej umożliwiającymi rozszerzenie do 640 KB);
– 1 napęd dysku elastycznego 360 KB;
– 1 równoległy układ wyjściowy (standardowy dla drukarki);
– standardowa klawiatura (83 klawisze);
– monitor monochromatyczny wraz z kartą grafiki typu Hercules lub monitor barwny z kartą typu CGA;
– opcjonalnie system operacyjny *MS–DOS 3.3*.

Jak już wspomniano, dla większości współczesnych zastosowań taka konfiguracja jest niewystarczająca, w innych przypadkach praca jest wprawdzie możliwa, ale możliwości wykorzystywanego oprogramowania są w dużym stopniu ograniczone, a konieczność bardzo częstej wymiany dyskietek z poszczególnymi częściami oprogramowania powoduje dużą uciążliwość pracy oraz dużą stratę czasu na przeładowywanie programów. Z tego też względu oferuje się rozszerzoną konfigurację podstawową, obejmującą najczęściej:
– dodatkowy napęd dysku elastycznego 360 KB lub 720 KB;
– pamięć RAM 512 KB;
– pamięć RAM 640 KB;
– dysk stały typu *Winchester* (minimum 20 MB);
– koprocesor arytmetyczny 8087;
– mysz;
– klawiaturę (101 klawiszy).

Obecnie można zaryzykować stwierdzenie, że minimalny, "rozsądny" zestaw mikrokomputera klasy XT do niedużych zastosowań, powinien zawierać podaną konfigurację podstawową z dodatkowym napędem dysku elastycznego 360 KB oraz pamięcią RAM 640 KB. W dalszej kolejności jest wskazane rozbudowanie systemu poprzez zainstalowanie dysku stałego (można wtedy zrezygnować z jednego napędu dysku elastycznego). Przykładowo, praca z procesorem tekstu *WordPerfect 4.2* jest możliwa (choć uciążliwa) bez dysku stałego, (nowsza wersja *5.0* ma aż 12 dyskietek instalacyjnych), a jego wykorzystanie w systemie z dyskami elastycznymi 360 KB bez dysku stałego nie jest praktycznie możliwe.

b) dla komputera klasy AT
– procesor 80286, 12/8 MHz (tzw. *turbo* – przełączalny);
– 512 KB pamięci RAM (z podstawkami na płycie głównej umożliwiającymi rozszerzenie do 640 KB, 1 MB lub więcej);
– 1 napęd dysku elastycznego 1,2 MB;
– 1 równoległy układ wyjściowy (standardowy dla drukarki);
– standardowa klawiatura (84 klawisze);
– monitor monochromatyczny wraz z kartą grafiki typu Hercules lub monitor barwny z kartą typu EGA;
– zegar czasu rzeczywistego z zasilaniem bateryjnym;
– opcjonalnie system operacyjny MS–DOS 3.3 (możliwy OS/2, Xenix).

Zestaw taki, choć oferowany w materiałach firmowych, nie ma praktycznego znaczenia bez dysku stałego typu *Winchester* o pojemności minimum 20 MB, a najlepiej

40 MB lub więcej. Ważnym parametrem, decydującym o przydatności danego dysku stałego do realizacji określonych zadań, jest także jego szybkość. Dalsze proponowane rozszerzenia to:
- napęd dysku elastycznego 360 KB, 1,2 MB lub 1,44 MB;
- pamięć RAM do 640 KB, 1 MB, 2 MB lub więcej;
- pakiety we–wy szeregowych, dodatkowych równoległych, w różnych zestawieniach;
- klawiatura (101 klawiszy);
- koprocesor arytmetyczny 80287;
- mysz;
- *streamer*.

W przypadku profesjonalnych zastosowań specjalistycznych, przy założeniu że dany zestaw jest przewidziany do realizacji ściśle określonego zadania i nie jest komputerem "do wszystkiego" obsługiwanym przez hobbystę–kolekcjonera oprogramowania, wybór właściwej konfiguracji sprzętu jest najczęściej jednym z ostatnich elementów analizy, którą należy przeprowadzić, począwszy od:
- precyzyjnego określenia tego, czego oczekujemy od kompletowanego systemu;
- przeprowadzenia rozeznania wśród dostawców oprogramowania; przeanalizowania możliwości uzyskiwanych poprzez zastosowanie różnych pakietów oprogramowania nie tylko pod kątem spełniania określonych wcześniej oczekiwań, ale także oferowanych możliwości rozbudowy, dostępności nowych wersji oprogramowania danej firmy, dostępności serwisu, możliwości uczestnictwa w szkoleniu prowadzonym przez firmę (lub inne specjalistyczne firmy) itp.;[1]
- określenia kwoty, jaką możemy przeznaczyć na zakup oprogramowania oraz niezbędnych dla nas specjalistycznych urządzeń we–wy (we–wy cyfrowe i analogowe, plotery, drukarki laserowe i inne).

Z przeprowadzonej analizy wynika dopiero, który zestaw mikrokomputera będzie dla nas odpowiedni.[2] Dla tego typu zastosowań cena samego mikrokomputera najczęściej będzie tylko pewną częścią ceny całego systemu (sprzęt i oprogramowanie) i decyzja wyboru takiej czy innej klawiatury, koprocesora itp. będzie mniej istotna.

Dla przykładu określimy ogólne konfiguracje systemów mikrokomputerowych, przeznaczonych do wspomagania projektowania układów cyfrowych oraz wspomagania prac wydawniczych.

Przykład 1 – komputerowe wspomaganie projektowania układów cyfrowych (przykład *CAD*)

Specjalistyczne urządzenia we–wy
- plotter o formacie A3 z czterema kolorami (lub więcej);
- monitor kolorowy o wymiarze rastrowym ekranu 800 × 600, ewentualnie 640 × 480 VGA);

---

[1] Ogólnie możemy powiedzieć, że nie powinny nas interesować firmy, które "nie opiekują się" swoimi klientami, posiadającymi ich produkty. Dotyczy to zresztą w równym stopniu oprogramowania i sprzętu.

[2] Zakładamy w dalszych rozważaniach, że typ mikrokomputera jest zdeterminowany do mikrokomputera zgodnego z IBM PC/XT/AT. W warunkach normalnie funkcjonującego rynku przyjęcie z góry takiego założenia jest niewłaściwe.

– dowolna, niezbyt wolna drukarka o szerokości wałka 15", pracująca w trybie graficznym.

Jednym z największych i funkcjonalnie najpełniejszych (w zakresie oprogramowania) systemów tego typu, przeznaczonych do pracy na mikrokomputerze klasy IBM PC, jest pakiet PC–CAD (dostępny także w kraju). Umożliwia on między innymi:
– projektowanie logiczne oraz symulowanie zaprojektowanych układów;
– projektowanie obwodów drukowanych;
– sporządzanie listy połączeń, listy odwiertów, elementów;
– sporządzanie dokumentacji w bardzo szerokim zakresie (z uwzględnieniem gospodarki magazynowej, zamówień na elementy itp.);
– tworzenie zbiorów dyskowych, służących do sterowania urządzeniami technologicznymi (montaż projektowanych układów cyfrowych).

Podstawą systemu powinien być możliwie duży i szybki zestaw mikrokomputera z dużym i szybkim dyskiem typu *Winchester*. Wymagania te wynikają z: konieczności operowania bardzo dużymi zbiorami[1], częstych transmisji dużej ilości danych pomiędzy dyskiem a pamięcią operacyjną oraz konieczności wykonywania przez procesor wielu złożonych obliczeń. Odpowiednim byłby więc mikrokomputer
– klasy AT (lub lepiej *super* AT z procesorem 80386) koniecznie z koprocesorem arytmetycznym;
– z dyskiem stałym typu *Winchester* 80 MB (lub więcej) o czasie dostępu 28 ms (lub mniejszym);
– z pamięcią RAM 2 MB (lub więcej) – do szybkiego buforowania;
– z myszą;
– ze *streamerem*.

Przykład 2 – wspomaganie pracy wydawniczej

Specjalistyczne urządzenia we–wy
– drukarka laserowa; zastosowanie typu zależy od możliwości finansowych użytkownika oraz od tego, czy będzie ona używana do przygotowywania gotowych matryc, czy też wyłącznie do wydruków kontrolnych (jeżeli przewidujemy automatyczny fotoskład);
– monitor; raczej monochromatyczny, ale o dużym rozmiarze rastrowym ekranu – 800 × 600 lub więcej.

W zakresie oprogramowania, najbardziej znane w kraju pakiety to PageMaker 3.0, pracujący w środowisku *Windows* oraz Ventura Publisher. Wybór właściwego pakietu uzależnia się od tego, jakiego rodzaju prace wydawnicze będą wykonywane. Z pewnym uproszczeniem można powiedzieć, że zastosowanie pakietu PagerMaker 3.0 będzie właściwsze przy pracach nad wydawaniem czasopism, artykułów przygotowanych na konferencje naukowo–techniczne itp. Umożliwia on także wykonanie niewielkich (najczęściej kontrolnych) wydruków przy pomocy normalnej igłowej drukarki mozaikowej. Przy pracach nad wydawaniem książek,

[1] Pakiet PC–CAD zajmuje (w zależności od wersji) na dysku ponad 15 MB.

właściwsze będzie zastosowanie Ventura Publisher[1], wówczas jednak można korzystać wyłącznie z drukarki laserowej. Wydruki na drukarce mozaikowej są złej jakości, a czas drukowania jest bardzo długi. Ponieważ programy te służą do ostatecznego przygotowania formy poszczególnych stron (wraz z rysunkami) i nie ma w nich możliwości rozbudowanej edycji tekstu, konfigurowany system powinien również zawierać wszystkie najbardziej znane procesory tekstów, takie jak: WordPerfect, Word, WordStar i inne. Należy założyć, że autorzy wydawanych tekstów będą dostarczać materiał przygotowany pod dowolnym, ogólnie akceptowanym procesorem tekstu. Zarówno w PagerMaker 3.0, jak i w Ventura Publisher są akceptowane zbiory z tekstami przygotowanymi pod wymienionymi wyżej programami edycyjnymi oraz z tekstami w "czystym" kodzie ASCII.

Za podstawę całego systemu należy przyjąć mikrokomputer
- klasy AT (koprocesor arytmetyczny mało istotny);
- z dyskiem stałym typu *Winchester* co najmniej 40 MB, możliwie szybkim;
- z pamięcią RAM 2 MB do szybkiego buforowania dużych zbiorów;
- z myszą;
- z drukarką 15" mozaikową 24-igłową,
- ze *streamerem*.

[1] Większość książek z serii "Mikrokomputery", a także książeczki z serii "PPP" i "PKE" przygotowuje się przy pomocy tego programu.

Literatura

1. Дао Л.: Програмирование микропроцесора 8088. МИР, Москва 1988.
2. Grabowski J., Koślacz St.: *Podstawy i praktyka programowania mikroprocesorów*. WNT, Warszawa 1987, wyd. II.
3. *IBM Personal Computer AT Technical Reference*. Copyright International Business Machines Corporation, USA 1984.
4. *IBM Personal Computer XT Technical Reference*. Copyright International Business Machines Corporation, USA 1983.
5. Исида Х.: Програмирование для микрокомпютеров. МИР, Москва 1988.
6. Jobst, Lutz, Sedler: *Intel 16/32 Bit Assemblerhandbuch*. Intereset – Verlag GmbH 1988.
7. Kozdrowicz T.: *IBM PC i PC DOS*. WKŁ, Warszawa 1988.
8. Lin Yu–Ch., Gibson G.A.: *Microcomputer systems – The 8086/8088 family*. Prentice – Hall Inc., Englewood Cliffs 1984.
9. Madej D., Marasek K., Kuryłowicz K.: *Komputery osobiste*. WKŁ, Warszawa 1987.
10. Małysiak H., Pochopień B., Podsiadło P., Wróbel E.: *Modułowe systemy mikrokomputerowe*. WNT, Warszawa 1990, wyd. II zmienione.
11. *Microsystem Components Handbook. Microprocessors and Pheripherials* – vol 1 and 2. Intel Corporation, Santa Clara 1985.
12. *Microsoft Macro Assembler 5.0 Reference*. Microsoft Corporation, USA and Canada 1987.
13. Misiurewicz P.: *Układy mikroprocesorowe*. WNT, Warszawa 1983.
14. *80286 Programmer's Reference Manual*. Intel Corporation, Santa Clara 1986.
15. Морисита И.: Аппаратные средцтва микро ЭВМ. МИР, Москва 1988.
16. *MS–OS/2 Programmer's Guide. Microsoft Operating System/2*. Software Development Kit. Microsoft Corporation, USA 1987.
17. *80286 System Software Writer's Guide*. Intel Corporation, Santa Clara 1986.
18. Wait M., Morgan Ch. L.: *Graphics Primer for the IBM PC*. Osborne/Mc Graw–Hill Berkeley, California 1983.
19. Wróbel E.: *Asembler 8086/8088*. WNT, Warszawa 1989.

AGENCJA
OBROTU
OPROGRAMOWANIEM

* *prowadzi:*
Rejestr Ochrony Oprogramowania

* *wydaje:*
katalog oprogramowania

* *pośredniczy:*
w sprzedaży oprogramowania

* *oferuje:*
licencjonowane oprogramowanie zagraniczne

* *proponuje:*
reklamę w książkach WNT

Nasz adres:
ul. Mazowiecka 2/4
00-048 Warszawa
pok. 206

Tel. 26-72-71
wew. 304
Fax 26-82-93

WYDAWNICTWA NAUKOWO-TECHNICZNE
ul. Mazowiecka 2/4, 00-048 Warszawa
tel. 26-72-71 do 79
Dział Upowszechniania i Sprzedaży
tel. 27-56-87

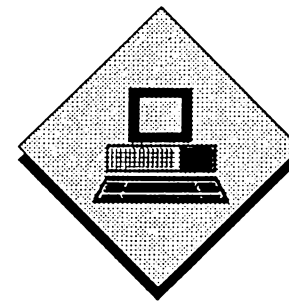
WNT Warszawa 1992
Wyd. I. Format B5
Ark. wyd. 20,1. Ark. druk. 13,75.
Symbol Et/82504/WNT
Druk: Szczecińskie Zakłady Graficzne
Zam. nr 600/11.10/91/II

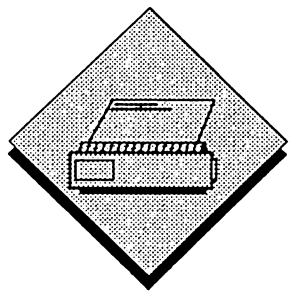
Z myślą o Tobie

QUMAK INTERNATIONAL ®

SPRZĘT KOMPUTEROWY

- COMPAQ
- SAMSUNG
- CMP



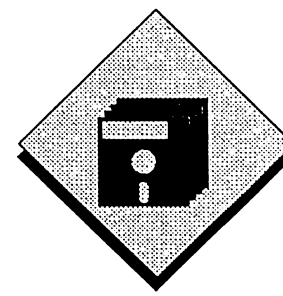


URZĄDZENIA PERYFERYJNE

- HEWLETT-PACKARD
- SEIKOSHA

OPROGRAMOWANIE

- MICROSOFT
- BORLAND
- NOVELL



Jesteśmy bezpośrednim, autoryzowanym partnerem handlowym tych firm. W naszej ofercie znajdziesz szeroki wybór ich produktów. Wszystko z myślą o Tobie. Zadzwoń, napisz lub przyjedź nawet dzisiaj. Zapraszamy.

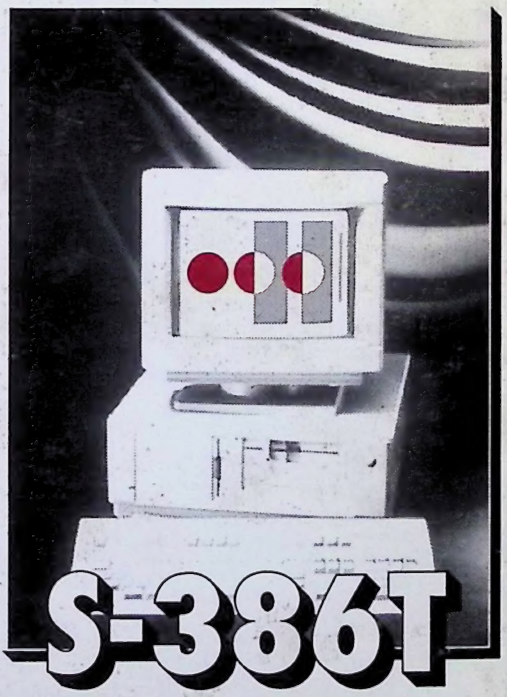
QUMAK INTERNATIONAL ®

ul. Szlak 65 31-153 KRAKÓW sklep firmowy
tel. (12) 22-51-44 wewn. 381, 388 ul. Poselska 24
tlx. 326356 qumak pl fax. 22-06-89 tel. (12)22-38-86

COMPUTER SYSTEM

ENTER 01/92 TEST

...niewątpliwie jednak testowany przez nas



to sprzęt wysokiej jakości!

TECHNOLOGIA XXI wieku w 1992 roku

Hyundai Electronics - wiodąca firma dalekowschodnia w dziedzinie przesyłania i przetwarzania informacji.
Od komputerów osobistych, poprzez laptopy i notebooki, po Sparc™Workstation.
Wielozadaniowe centrale telefoniczne i nowoczesne telefaxy.
Drukarki i monitory.
Zgodność z wymaganiami standardu Novell, potwierdzona certyfikatami.
1.5-roczna gwarancja i perfekcyjne testy producenta.
Nasze atuty to wszechstronność i umiejętność rozwiązania każdego problemu naszego Klienta.

HYUNDAI
SELKO INDUSTRIES LTD.

00-762 Warszawa, ul. Belwederska 20/22, tel.0-22/41 40 05, 41 19 77, fax 41 36 08

© Studio Q

ISBN 83-204-1461-X